LATHAM & WATKINS LLP
Mark A. Flagel (Bar No. 110635)
Ryan E. Hatch (Bar No. 235577)
Jessica C. Kronstadt (Bar No. 267959)
mark.flagel@lw.com
ryan.hatch@lw.com
jessica.kronstadt@lw.com
355 South Grand Avenue
Los Angeles, California 90071-1560
Telephone: (213) 485-1234
Facsimile: (213) 891-8763

LATHAM & WATKINS LLP
Dean G. Dunlavey (Bar No. 115530)
dean.dunlavey@lw.com
650 Town Center Drive - 20th Floor
Costa Mesa, California 92626-1925
Telephone: (714) 540-1235
Facsimile: (714) 755-8290

Attorneys for Plaintiff
A10 NETWORKS, INC.

UNITED STATES DISTRICT COURT

CENTRAL DISTRICT OF CALIFORNIA

A10 NETWORKS, INC., a Delaware Corporation,

Plaintiff,

v.

BROCADE COMMUNICATIONS SYSTEMS, INC., a Delaware Corporation, and F5 NETWORKS, INC., a Washington Corporation,

Defendants.

CASE NO. SACV11-01378 JST (ANx)

COMPLAINT FOR PATENT INFRINGEMENT

DEMAND FOR JURY TRIAL

Plaintiff A10 Networks, Inc. ("A10"), for its Complaint against Defendants Brocade Communications Systems, Inc. ("Brocade") and F5 Networks, Inc. ("F5") alleges as follows:

JURISDICTION AND VENUE

1. This is an action for patent infringement, under the patent laws of the United States, 35 U.S.C. § 271 et seq. This Court has jurisdiction under 28 U.S.C. §§1331 and 1338(a).

2. Venue is proper within this judicial district under 28 U.S.C. §§1391(b) and (c) because Brocade has offices in this district, the Defendants sell and distribute infringing products within this district, and Defendants are subject to personal jurisdiction here.

THE PARTIES

3. Plaintiff A10 is a California corporation, with its principal place of business at 2309 Bering Drive, San Jose, California. A10 also has an office within this district in Laguna Hills.

4. Defendant Brocade is a Delaware corporation, with its principal place of business at 1745 Technology Drive, San Jose, California. Brocade sells and distributes infringing products within this district, and is believed to maintain offices within this district in Irvine.

5. Defendant F5 is a Washington corporation, with its principal place of business at 401 Elliott Avenue West, Seattle, Washington. F5 sells and distributes infringing products within this district, and is believed to maintain an office in San Jose, California.

///

///

///

A10'S PATENT RIGHTS

6. A10 was founded in 2004 and provides innovative networking and security solutions that help organizations accelerate, optimize and secure their applications. A10's flagship product line, the AX Series, is a family of high performance server load balancers that are the recognized industry leader in high performance traffic management and application delivery controllers. The AX Series' Advanced Core Operating System (ACOS) architecture has garnered numerous awards and is revolutionary by market standards.

7. On November 21, 2006, the United States Patent and Trademark Office issued United States Patent No. 7,139,267 (the "'267 Patent"), entitled "System and Method of Stacking Network Switches," a copy of which is attached hereto as Exhibit A. The '267 Patent is generally directed at the use of refresh packets in the context of synchronizing the forwarding databases of network switches.

8. A10 is the owner by assignment of the '267 Patent, and has the right to sue for infringement of the '267 Patent.

9. On June 26, 2007, the United States Patent and Trademark Office issued United States Patent No. 7,236,491 (the "'491 Patent"), entitled "Method and Apparatus for Scheduling For Packet-Switched Networks," a copy of which is attached hereto as Exhibit B. The '491 Patent is generally directed at the use of priority queuing and scheduling in the context of packet transmission.

10. A10 is the owner by assignment of the '491 Patent, and has the right to sue for infringement of the '491 Patent.

COUNT I

(PATENT INFRINGEMENT – '267 Patent, Against All Defendants)

11. A10 repeats and realleges the allegations set forth in paragraphs 1-10, above, as though fully set forth hereat.

12. A10 is informed and believes that a reasonable opportunity for further investigation and discovery will confirm that Brocade has been and is infringing the '267 Patent directly and indirectly by making, using, selling and/or offering for sale products, including at least its FastIron SX, CX, LS and GS, NetIron XMR, MLX and MLXE, and BigIron RX series products, that practice or enable the practice of inventions claimed in one or more of the claims of the '267 Patent.

13. A10 is informed and believes that a reasonable opportunity for further investigation and discovery will confirm that F5 has been and is infringing the '267 Patent directly and indirectly by making, using, selling and/or offering for sale products, including at least its BIG-IP products, that practice or enable the practice of inventions claimed in one or more of the claims of the '267 Patent.

14. Defendants' acts of infringement have caused, and will continue to cause, substantial and irreparable injury to A10 and its rights.

COUNT II

(PATENT INFRINGEMENT – '491 Patent, Against All Defendants)

15. A10 repeats and realleges the allegations set forth in paragraphs 1-10, above, as though fully set forth hereat.

16. A10 is informed and believes that a reasonable opportunity for further investigation and discovery will confirm that Brocade has been and is infringing the '491 Patent directly and indirectly by making, using, selling and/or offering for sale products, including at least its FastIron SX, NetIron XMR, MLX and MLXE, SeverIron ADX, and BigIron RX series products, that practice or enable the practice of inventions claimed in one or more of the claims of the '491 Patent.

17. A10 is informed and believes that a reasonable opportunity for further investigation and discovery will confirm that F5 has been and is infringing

the '491 Patent directly and indirectly by making, using, selling and/or offering for sale products, including at least its BIG-IP products, that practice or enable the practice of inventions claimed in one or more of the claims of the '491 Patent.

18. Defendants' acts of infringement have caused, and will continue to cause, substantial and irreparable injury to A10 and its rights.

REQUEST FOR RELIEF

WHEREFORE, A10 seeks judgment against Brocade and F5, and each of them, as follows:

1. A preliminary and permanent injunction, enjoining Defendants and their agents, servants, employees and all those in privity with them from making, using, selling or offering for sale any product that infringes or enables infringement of the '267 Patent, from contributing to the infringement of that patent, or from inducing the infringement of that patent;

2. A preliminary and permanent injunction, enjoining Defendants and their agents, servants, employees and all those in privity with them from making, using, selling or offering for sale any product that infringes or enables infringement of the '491 Patent, from contributing to the infringement of that patent, or from inducing the infringement of that patent;

3. An award of damages adequate to compensate A10 for Defendants' unlawful infringement, in an amount to be shown according to proof at trial, but in no event less than a reasonable royalty;

4. An award of A10's costs of suit, including reasonable attorneys' fees, pursuant to U.S.C. § 285 or as otherwise permitted by law; and

5. Such other and further relief as to the Court seems just and proper.

///

///

JURY DEMAND

Pursuant to Rule 38(b) of the Federal Rules of Civil Procedure, A10 hereby demands trial by jury of all issues so triable that are raised herein or which hereafter may be raised in this action.

DATED: September 9, 2011

Respectfully submitted,

LATHAM & WATKINS LLP
Mark A. Flagel
Dean G. Dunlavey
Ryan E. Hatch
Jessica C. Kronstadt

By
Mark A. Flagel
Attorney for Plaintiff
A10 Networks, Inc.

LA\2293726.5

EXHIBIT A

US007139267B2

(12) **United States Patent**
Lu et al.

(10) **Patent No.: US 7,139,267 B2**
(45) **Date of Patent: Nov. 21, 2006**

(54) **SYSTEM AND METHOD OF STACKING NETWORK SWITCHES**

(75) Inventors: **Kuo-Cheng Lu**, Hsin-Chu (TW); **Hung-Kuang Chen**, Hsin-Chu Hsien (TW)

(73) Assignee: **Industrial Technology Research Institute**, Hsinchu (TW)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1010 days.

(21) Appl. No.: **10/087,761**

(22) Filed: **Mar. 5, 2002**

(65) **Prior Publication Data**

US 2003/0169734 A1 Sep. 11, 2003

(51) **Int. Cl.**
H04L 12/50 (2006.01)

(52) **U.S. Cl.** **370/386**; 370/389; 370/400

(58) **Field of Classification Search** 370/386, 370/382, 379, 428
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,732,041	A	3/1998	Joffe	
5,909,686	A *	6/1999	Muller et al.	707/104.1
6,058,116	A	5/2000	Hiscock et al.	
6,813,268	B1 *	11/2004	Kalkunte et al.	370/392
2002/0124107	A1 *	9/2002	Goodwin	709/242
2005/0030948	A1 *	2/2005	Wyatt	370/392
2005/0063313	A1 *	3/2005	Nanavati et al.	370/252

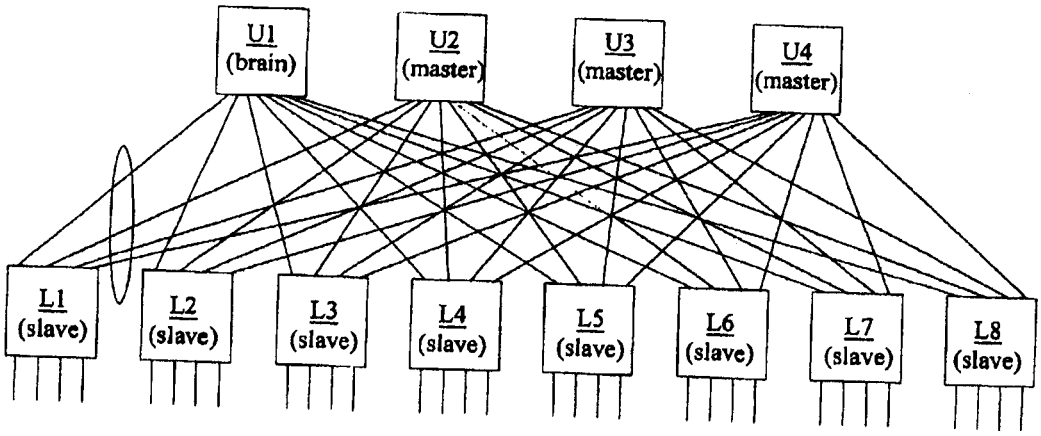
* cited by examiner

Primary Examiner—Chau Nguyen
Assistant Examiner—Christopher Grey
(74) *Attorney, Agent, or Firm*—Finnegan, Henderson, Farabow, Garrett & Dunner, L.L.P.

(57) **ABSTRACT**

A network switch system that includes a plurality of first-level switches operating in a slave mode, the first-level switches providing a plurality of local ports for receiving and sending network packets, and a plurality of second-level switches operating in one of brain mode or master mode, wherein, the first-level switches includes a plurality of upward ports connecting to the second-level switches, each of the first-level switches and the second-level switches having a forwarding database, wherein the first-level switches sends the refresh packets to the second-level switches for synchronizing the forwarding databases of the second-level switches, wherein the second-level switches providing packet communications among the first-level switches, and wherein a second-level switch operating in the brain mode providing refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

36 Claims, 9 Drawing Sheets



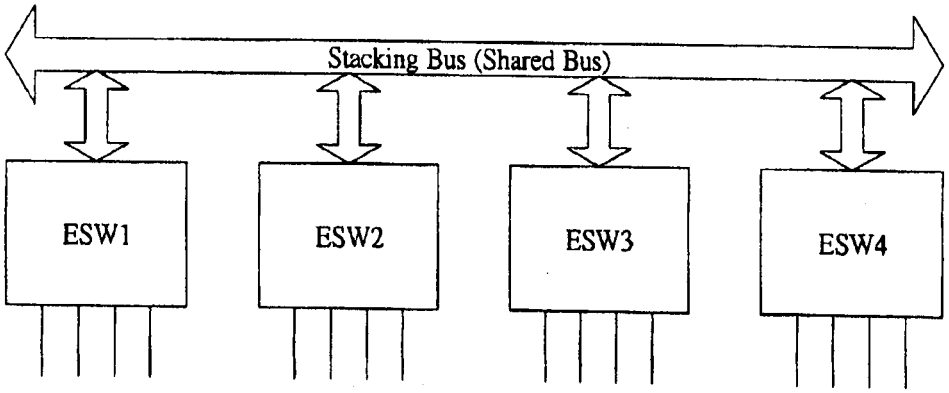


Fig. 1 (prior art)

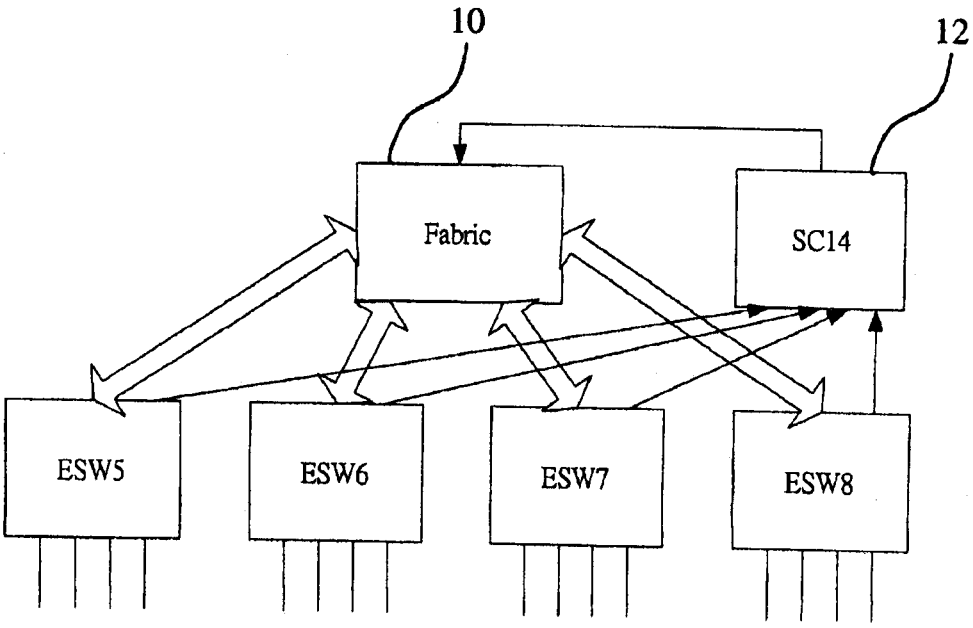


Fig. 2 (prior art)

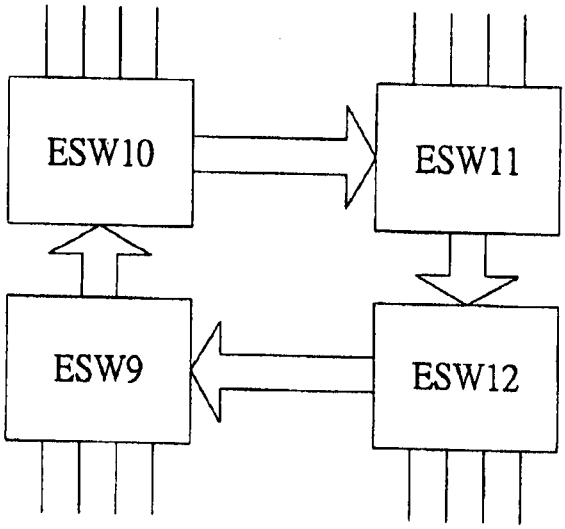


Fig. 3 (prior art)

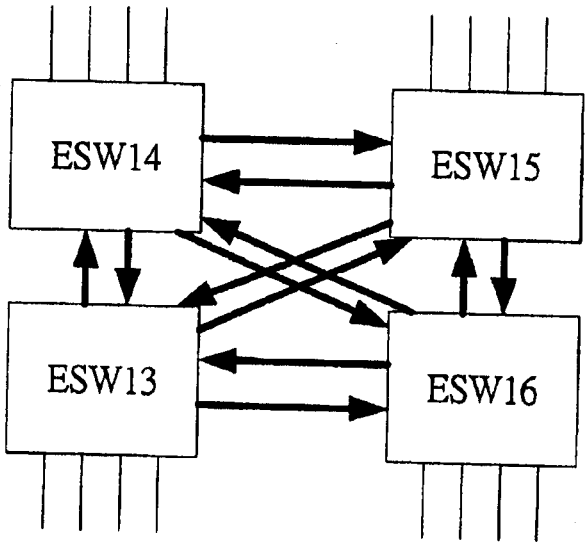


Fig. 4 (prior art)

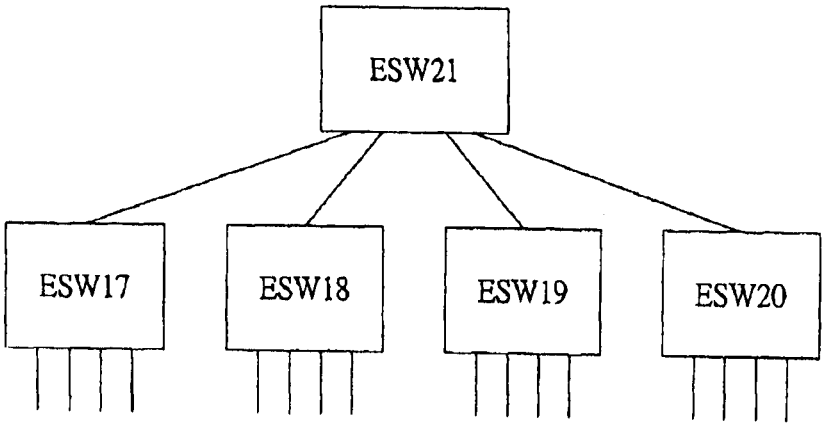


Fig. 5 (prior art)

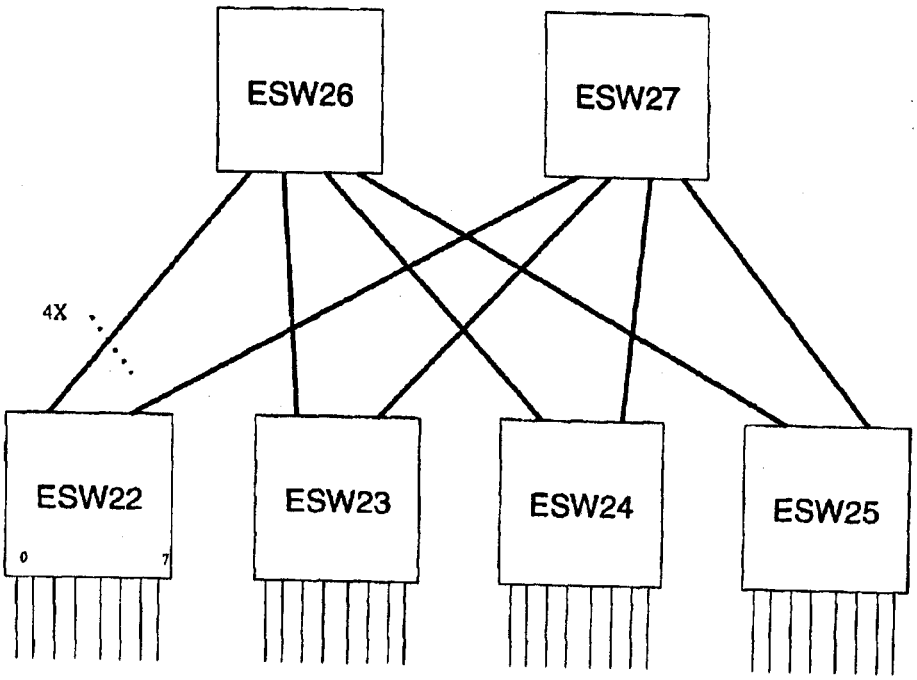


Fig. 6 (prior art)

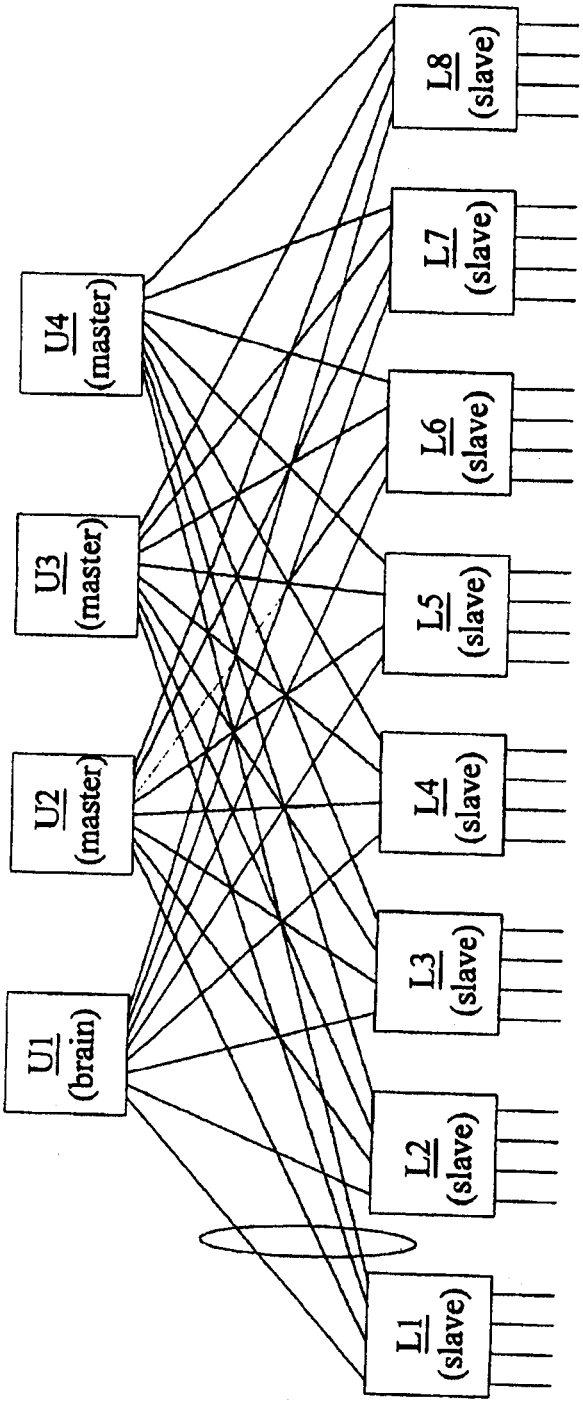

U1 (brain)
U2 (master)
U3 (master)
U4 (master)
L1 (slave)
L2 (slave)
L3 (slave)
L4 (slave)
L5 (slave)
L6 (slave)
L7 (slave)
L8 (slave)

Fig. 7

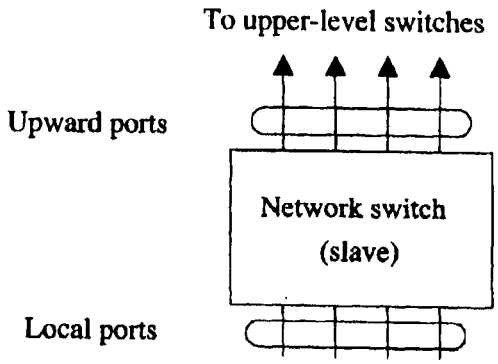

To upper-level switches
Upward ports
Network switch
(slave)
Local ports

Fig. 8

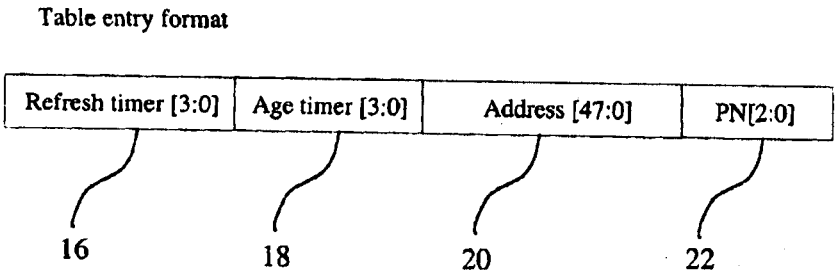


Table entry format
Refresh timer [3:0]
Age timer [3:0]
Address [47:0]
PN[2:0]
16
18
20
22

Fig. 9

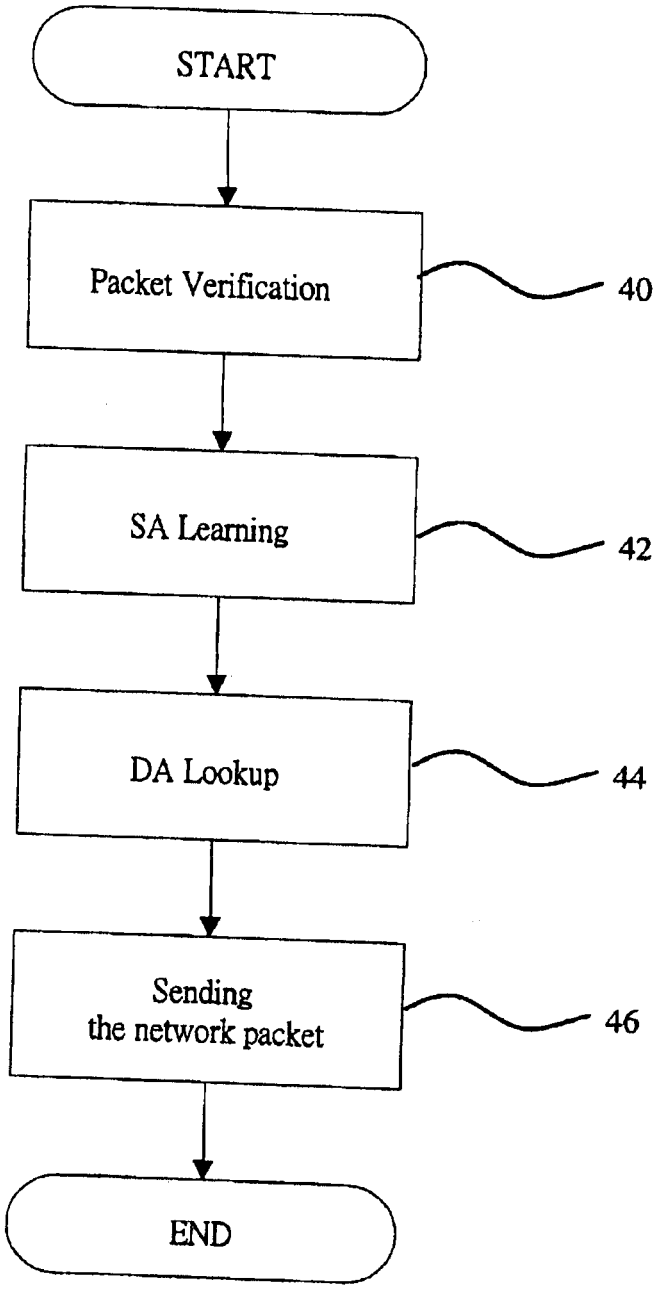

START
Packet Verification
40
SA Learning
42
DA Lookup
44
Sending
the network packet
46
END

Fig. 10

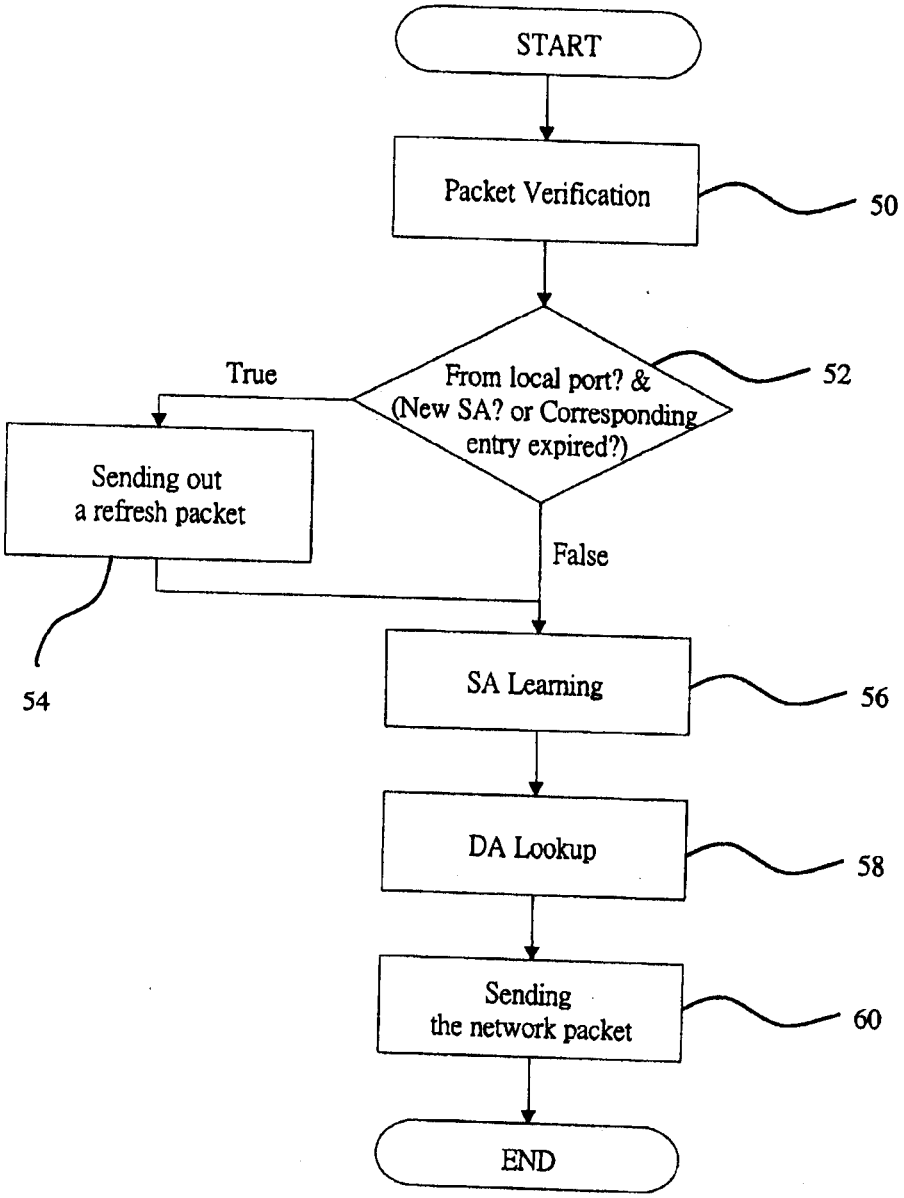

START
Packet Verification
50
From local port? &
(New SA? or Corresponding
entry expired?)
52
True
Sending out
a refresh packet
54
False
SA Learning
56
DA Lookup
58
Sending
the network packet
60
END

Fig. 11

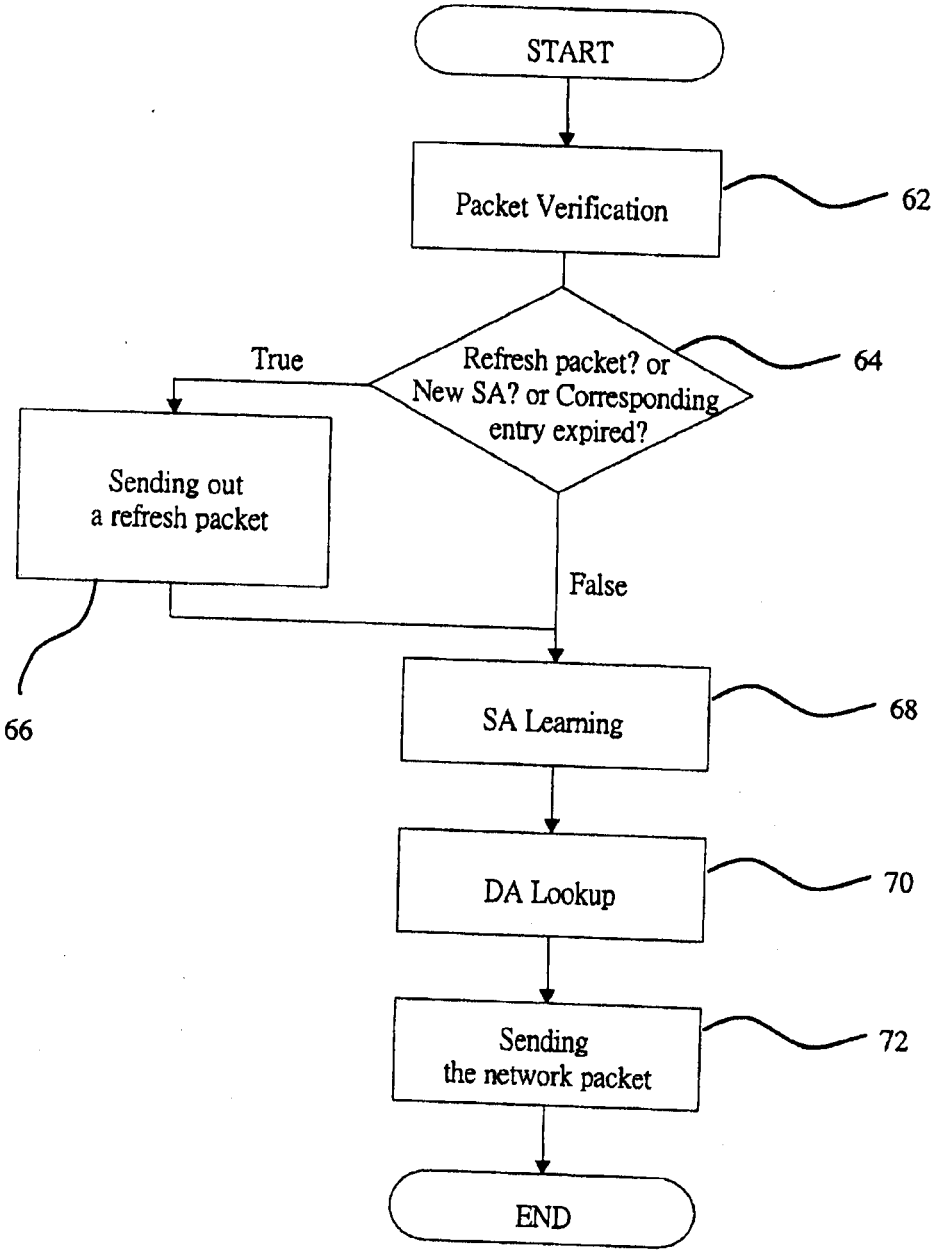

START
Packet Verification
62
Refresh packet? or New SA? or Corresponding entry expired?
64
True
Sending out a refresh packet
66
False
SA Learning
68
DA Lookup
70
Sending the network packet
72
END

Fig. 12

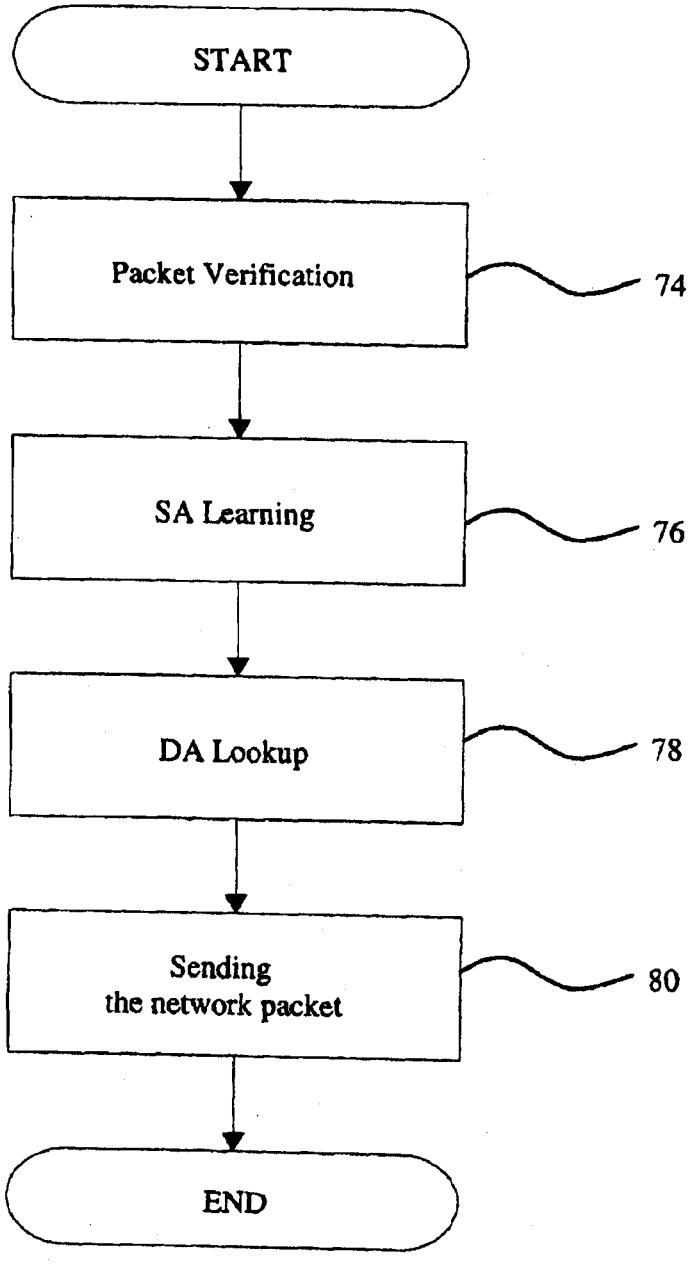

START
Packet Verification
74
SA Learning
76
DA Lookup
78
Sending
the network packet
80
END

Fig. 13

SYSTEM AND METHOD OF STACKING NETWORK SWITCHES

DESCRIPTION OF THE INVENTION

1. Field of the Invention

This invention pertains in general to systems and methods for stacking network switches, and more particularly, to systems and methods for stacking network switches with forwarding database synchronization.

2. Background of the Invention

A network system generally provides the exchange of information among multiple ends. Taking a computer network system as an example, the system provides the exchange of information among multiple nodes or segments. Each node or segment may contain one or more terminals, which may be a server, personal computer ("PC"), input device, or output device, such as a printer or plotter. In general, network switches manage and regulate traffic among the nodes or segments of a network system to enable efficient exchange of data and reduce network traffic congestion by directing packets to their designated destinations.

In an Ethernet network system, an Ethernet switch integrated circuit ("IC") is an Application Specific Integrated Circuit ("ASIC") that provides the capability to manage and regulate network traffic. The need for systems having different numbers of ports has driven the development of various types of specifications for Ethernet switch ICs. For example, systems designed for high-speed Ethernet operations frequently integrate a significant number of ports in a single chip to promote efficiency and traffic-handling capabilities. This type of chips typically require significant memory bandwidth and, as a result, substantial development time and considerable production cost.

Under the general operating procedures of Ethernet switching, an Ethernet packet records the destination and source addresses in the first 12 bytes of the packet data. When an Ethernet switch receives a packet, the switch retrieves the destination address and looks up the output port for the destination in a forwarding database that provides a corresponding port to each address. The switch then sends the packet to the identified destination port. A network switching system may create the forwarding database by registering the source address and port information of each incoming packet to the forwarding database. IEEE 802.1D, a standard on Media Access Control ("MAC") bridges, provides the operation procedures for creating forwarding databases and transmitting and receiving of network packets. A "learning process" for a network switch systems means the aforementioned operation of registering source address and port information to a forwarding database. The process enables an Ethernet switch to send incoming packets to their destination ports based on the registered information in the forwarding database.

Each Ethernet switch has only a limited number of ports and manages network traffic among a limited number of nodes or segments. For an Ethernet switch system to provide more ports, the system frequently has to combine, or "stack," two or more Ethernet switches and provides interconnections between the Ethernet switches to enable coordinated operation of the combined system. stacking can categorized into one of four types: shared bus, switching fabric, ring-bus, and interleaved interconnection structures.

FIG. **1** illustrates a functional block diagram of a system with a known shared bus structure. Referring to FIG. **1**, each of ESW**1**, ESW**2**, ESW**3**, and ESW**4** represents an Ethernet switch, and each contains a packet memory. When an

Ethernet switch receives an input packet, the Ethernet switch first makes a forwarding decision to determine the delivery path of the packet. The Ethernet switch stores the input packet temporarily into the packet memory of the switch. If the destination port of the input packet is within the same local area managed by the Ethernet switch, the switch sends the input packet directly through one of the output ports of the switch.

Conversely, for an input packet having a destination port outside the local area of the Ethernet switch, the Ethernet switch first sends a request through the shared bus to request that the destination Ethernet switch have a packet buffer available. The Ethernet switch then sends the packet through the shared bus. After the destination Ethernet switch receives the full packet, the Ethernet switch releases the buffer that stores the packet. The destination Ethernet switch then stores the input packet in its packet memory and sends the packet through one of its output ports.

The second known stacking approach, the switching fabric structure, may be further divided into a shared-memory switching fabric and crossbar switching fabric. A network switch with a shared-memory switching fabric structure allows all network ports to write arriving data, such as packets from input ports, into a shared memory. An output port then reads out the packet after the switch system makes a forwarding decision. However, the write-read operations require repetitive access to a significant amount of centralized shared-memories. The requirement limits the expandability of the system because the shared memory block of the system has a limited bandwidth and storage space. As an illustrative example, the shared-memory switching fabric of an Ethernet port of 16 Gigabit requires a bandwidth of 32 Gigabit to enable the write-read operations. The requirement on large bandwidth and memory size poses a significant challenge for modern memory design of Ethernet switches with expanded bandwidths.

FIG. **2** illustrates a functional block diagram of a network switch system with a crossbar switching fabric structure **10**. Comparing with the shared-memory switching fabric structure, a crossbar switching fabric structure seeks to increase the capability of a switching fabric. The system in FIG. **2** provides an input queue (not shown) to store packets at the input end of each port. A crossbar scheduler **12** then provides an output port upon request. Packets are sent through the switching fabric **10** that switches, or channels, the packets to an output port according to instructions from the scheduler **12**. Because this approach does not rely on a shared memory block to provide a buffer zone for packet exchange, it eliminates the reliance on memory bandwidth of the shared-memory switching fabric structure and therefore provides improved expandability. This type of switching fabric structure, however, requires a sophisticated scheduler to efficiently cooperate with the switch fabric and provide full throughput by effective delay control of packet transmissions. The requirement on scheduler capability poses substantial development and production costs.

As FIG. **2** illustrates, each of the Ethernet switches ESW**5**, ESW**6**, ESW**7**, and ESW**8** connects to the scheduler **12**. When an Ethernet switch, for example, ESW**5**, needs to send a packet to another Ethernet switch, for example, ESW**8**, ESW**5** sends a request to the scheduler **12** in order to have a point-to-point connection arranged by the scheduler **12**. ESW**5** then sends the packet as the switching fabric **10** provides a channel between the sender Ethernet switch ESW**5** and the receiving Ethernet switch ESW**8**.

FIG. **3** illustrates a functional block diagram of a network switch system with a ring-bus structure. Four Ethernet

switches, ESW9, ESW10, ESW11, and ESW12 connect sequentially and circularly for forming a ring structure. This system, due to its nature, often transmits a packet through one or more intermediate Ethernet switches in order to send a packet. For example, in order to send a packet from ESW9 to ESW12, the switch structure first sends the packet to ESW10, then ESW11, and finally ESW12. The process consumes additional processing resources of the switches and limits the throughput of the switching system. Allayer Communications and PMC-Sierra, Inc., both of the Unites States, provide Ethernet switch products that employ similar ring-stacking structures.

FIG. 4 illustrates a functional block diagram of a network switch system with an interleaved interconnection structure. Alexander illustrates, in U.S. Pat. No. 5,732,041, memory interface systems for providing and arbitrating access to a shared memory by multiple devices. As FIG. 4 illustrates, four Ethernet switches, ESW13, ESW14, ESW15, and ESW16, cross-connect with each other. Each of the four Ethernet switches has a packet memory for storing packets that are awaiting transmission.

The switching architectures described above all require special stacking buses or wirings to cross-connect the Ethernet switches, and have several major drawbacks. First, there is not a standard interface or protocol for the stacking buses among switches and therefore this structure requires a customized design according to the types of Ethernet switches. Second, the stacking buses cannot be used as ordinary network ports and thus would be useless as a stand-alone. Third, the stacking buses have limited bandwidth and this inherent limitation restricts the expandability of the network system. Fourth, the requirement of a customized design of the stacking buses increases the system cost.

Other than employing switching various fabrics, Ethernet switches may be connected through Ethernet ports in a two-level structure to form a switch system that provides more ports. FIG. 5 illustrates a functional block diagram of a two-level structure where four lower-level Ethernet switches, ESW17, ESW18, ESW19, and ESW20, are connected to an upper-level Ethernet switch ESW21. Whenever any of the lower level Ethernet switches ESW17, ESW18, ESW19, or ESW20 needs to send a packet, the lower level Ethernet switch first sends the packet to the upper level Ethernet switch ESW21, and the packet is then sent to the intended destination through ESW21. The two-level structure system requires no specific modification or re-design of Ethernet switches. As a result, Ethernet from different manufacturers or of different designs can be combined to provide a switch system with an increased number of ports.

The two-level structure system, however, is not a non-blocking structure because all packets from the ports of the Ethernet switches ESW17, ESW18, ESW19, and ESW20, pass through a single Ethernet switch ESW21. Without a non-blocking design, packet loss may occur when each port sends packets at full throughput to different output ports and the full traffic congests ESW21. The problem of packet loss comes from the limited bandwidth of the connections between the upper-level Ethernet switch ESW21 and the lower-level Ethernet switches ESW17, ESW18, ESW19, and ESW20. Nevertheless, because of its simplicity, this structure is frequently employed in systems that emphasize less on non-blocking features or traffic handling capacities.

SwitchCore AB of Sweden ("SwitchCore") has provided a different structure in its products. FIG. 6 illustrates a functional block diagram of a SwitchCore switch system, which follows the traditional connection structure that aims to provide load balancing and redundant capabilities in network planning, as described by Hiscock et al. in U.S. Pat. No. 6,058,116, titled "Interconnected trunk cluster arrangement." Referring to FIG. 6, each of lower level Ethernet Switches ESW22, ESW23, ESW24, and ESW25 has the same number of cascading and user ports, and the switch system is capable of achieving non-blocking. However, the switch system needs to synchronize the forwarding databases of the system to avoid unnecessary broadcasts of packets and potential sequencing disorder of packets. Because the SwitchCore system lacks a centralized forwarding database, the system instead generates several forwarding databases. The forwarding databases of upper-level Ethernet switches do not possess the same address-port information as those of the lower-level Ethernet switches. This non-uniformity of the forwarding databases may cause system operation instabilities. As an example, a first Ethernet switch might not have the source address and port information that has been registered to a second Ethernet switch. As a result, the first Ethernet switch has to broadcast a packet when the first Ethernet switch does not contain any forwarding port information for a particular packet. The broadcasting operation increases the number of broadcasting packets and thus additional network traffic. Transmission of the broadcasting packets therefore increases the load on the system and reduces the system throughput.

Referring again to FIG. 6, the SwitchCore approach enables lower-level Ethernet switches ESW22, ESW23, ESW24, and ESW25 to learn the source addresses and ports of only packets from the lower-level ports, but not packets from the cascaded ports connected to the upper-level Ethernet switches ESW26 and ESW27. Each of the lower-level Ethernet switches, therefore, only has the port information of the users of that switch, but not the port information of the users of other Ethernet switches. For example, the switch ESW22 possesses the port information of its own users, but not the port information of the users of the switch ESW24. When the lower-level Ethernet switch ESW22 receives a packet of unregistered destination address, the switch ESW22 sends the packet to an upper-level Ethernet switch, for example, ESW26, through the cascaded port between the two switches. The upper-level switch ESW26 then determines the destination port information and sends the packet to the destination lower-level switch, for example, ESW24. The SwitchCore approach, therefore, allows and necessarily requires the upper-level Ethernet switches to have uniform forwarding databases by notifying a system Central Processing Unit ("CPU") whenever the switch learns about new source address and port information, so that the system may register the information to the forwarding database of each of the upper-level Ethernet switches.

The SwitchCore approach therefore allows the use the same type of Ethernet switches to construct an expanded Ethernet switch system, and provides that the lower-level Ethernet switches only need to register the source address and port information from the local ports. In addition, only the upper-level Ethernet switches need to maintain comprehensive forwarding databases with uniform information by the operation of the system CPU. As a result, however, the SwitchCore approach requires the system CPU to constantly process source address and source port information and update all the forwarding databases of the upper-level switches, which represent a significant load on the CPU. Furthermore, when an upper-level Ethernet switch does not have the destination port information for a packet of an unregistered destination address, the upper-level switch must send the packet back to all of the lower-level Ethernet

switches, including the source lower-level Ethernet switch that delivers the packet. The returning operation occupies additional bandwidth and limits the availability of the system bandwidth to handle other network packets.

Without limiting the scope of the present invention, the foregoing paragraphs illustrate the background of the present invention with an exemplary Ethernet network system. The systems and methods of the present invention are also applicable to different types of network systems.

SUMMARY OF THE INVENTION

Accordingly, the present invention is directed to systems and methods for stacking network switches that substantially obviate one or more of the problems due to limitations and disadvantages of the related art.

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be apparent from the description, or may be learned by practice of the invention. The objectives and other advantages of the invention will be realized and attained by the systems and methods particularly pointed out in the written description and claims hereof, as well as the appended drawings.

To achieve these and other advantages, and in accordance with the purpose of the invention as embodied and broadly described, the present invention provides a network switch system that includes a plurality of network switches for providing an exchange of network packets, each of the network switches including a forwarding database, wherein the network switch system is capable of providing at least one refresh packet, upon receiving the network packets, to synchronize the forwarding databases of the plurality of network switches, and wherein each of the plurality of network switches registers the at least one refresh packet to the forwarding database of the network switch upon receiving the refresh packets.

In one aspect, the forwarding databases include at least one refresh timer in an address entry for recording the validity of a corresponding address entry in the forwarding databases of neighboring switches.

In another aspect, the forwarding databases include an address entry having an age timer for the address entry that records the validity of the address entry, an address for the address entry, and associated port information for the address.

In yet another aspect, the plurality of network switches includes a first switch and a second switch, each having a forwarding database, the first switch sending a refresh packet to a second switch when the first switch receives a network packet and the network packet containing address information that the forwarding database of the first switch has no corresponding address entry; or the network packet containing address information that a corresponding address entry in the forwarding database of the second network switch has expired.

In still another aspect, the plurality of network switches further includes a plurality of first-level switches having a plurality of upward ports operating in a slave mode, a plurality of second-level switches including a channeling switch, wherein each of the first-level switches is configured to connect to each of the second-level switches, wherein at least one second-level switch operates in a brain mode and at least one second-level switch operates in a master mode, wherein the first-level switches providing a plurality of local ports for receiving and sending network packets, the upward ports of the first-level switches connecting to the second-level switches, the first-level switches sending the refresh packets to the second-level switches for synchronizing the forwarding databases of the second-level switches, and wherein the second-level switches providing packet communications among the first-level switches, the second-level switch operating in the brain mode providing refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

The present invention also provides a network switch system that includes a plurality of first-level switches operating in a slave mode, the first-level switches providing a plurality of local ports for receiving and sending network packets, and a plurality of second-level switches operating in one of brain mode or master mode, wherein, the first-level switches includes a plurality of upward ports connecting to the second-level switches, each of the first-level switches and the second-level switches having a forwarding database, wherein the first-level switches sends the refresh packets to the second-level switches for synchronizing the forwarding databases of the second-level switches, wherein the second-level switches providing packet communications among the first-level switches, and wherein a second-level switch operating in the brain mode providing refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

In one aspect, one of the first-level switches sends a refresh packet to the second-level switches connected with the first-level switch when the first-level switch receives a network packet, and the packet contains address information that the forwarding database of the first-level switch has no corresponding address entry, or the network packet contains address information that a corresponding address entry in the forwarding databases of one of the second-level switches has expired.

In another aspect, the second-level switch operating in the brain mode sends a refresh packet to the first-level switches connected with the second-level switch if the second-level switch operating in the brain mode receives a network packet, and the network packet contains address information that the forwarding database of the second-level switch operating in the brain mode has no corresponding address entry, the network packet containing address information that corresponding address entries in the forwarding databases of the first-level switches have expired, or the second-level switch operating in the brain mode receives an incoming refresh packet from the first-level switches.

In yet another aspect, one of the second-level switches needs to send a refresh packet coming from a first-level initiating switch or containing the source address information of a network packet from a first-level source switch, the second-level switch sends the refresh packet to each of the first-level switches except the first-level initiating switch and the first-level source switch.

The present invention further provides a method for operating a network switch in a slave mode within a network switch system, the slave switch having a forwarding database that includes receiving a network packet, sending an outgoing refresh packet to neighboring switches when the network packet contains address information that the slave switch has no corresponding record in the forwarding database of the slave switch, or the network packet contains address information that corresponding address entries in the forwarding databases of the neighboring switches have expired, wherein the slave switch sends the outgoing refresh packet to synchronize the forwarding databases of the neighboring switches, registering the address information of the network packet to the forwarding database of the slave

switch, and registering the address information of an incoming refresh packet to the forwarding database of the slave switch upon receiving the incoming refresh packet.

In one aspect, the method further includes looking up the destination port of the network packet in the forwarding database of the slave switch, sending the network packet to the destination port, and broadcasting the network packet when the forwarding database of the slave switch has no corresponding destination port information for the network packet.

In another aspect, the forwarding database of the slave switch includes a refresh timer in an address entry for recording the validity of a corresponding address entry in the forwarding databases of the neighboring switches.

In still another aspect, the forwarding database of the slave switch includes an address entry containing an age timer for the address entry that records the validity of the address entry, an address for the address entry, and associated port information for the address.

The present invention additional provides a method of operating a network switch in a brain mode within a network switch system, the brain switch having a forwarding database that includes receiving a network packet, sending an outgoing refresh packet to neighboring switches when the network packet contains address information that the forwarding database of the brain switch has no corresponding address entry, the network packet contains address information that corresponding address entries in the forwarding databases of the neighboring switches have expired, or the brain switch receives an incoming refresh packet from the neighboring switches, wherein the brain switch sends the outgoing refresh packet to synchronize the forwarding databases of the neighboring switches, and registering the address information of the network packet to the forwarding database of the brain switch.

The present invention also provides a method of operating a network switch in a master mode within a network switch system, the master switch having a forwarding database that includes receiving an incoming network packet, registering address information of an incoming refresh packet to the forwarding database of the master switch upon receiving the incoming refresh packet, looking up the destination port of the incoming network packet in the forwarding database of the master switch, sending the incoming network packet to the destination port, and broadcasting the incoming network packet when the forwarding database does not have corresponding destination port information.

In one aspect, a neighboring switch of the master switch sends the refresh packet to the master switch when the neighboring switch receives an original network packet, and the original network packet contains address information that the forwarding database of the neighboring switch has no corresponding address entry, or the original network packet contains address information that a corresponding address entry in the forwarding databases of the master switch has expired.

In another aspect, the forwarding database of the master switch includes an address entry containing an age timer for the address entry that records the validity of the address entry, an address for the address entry, and associated port information for the address.

The present invention further provides a method for stacking network switches that includes providing a plurality of network switches for providing an exchange of network packets, providing a forwarding database for each of the network switches, providing at least one refresh packet to synchronize the forwarding databases of the plurality of network switches upon receiving the network packets, and registering the refresh packet for each of the plurality of network switches to the forwarding databases.

In one aspect, the method further includes providing a first switch and a second switch, providing a forwarding database for the first switch and the second switch, and the first switch sending a refresh packet to a second switch when the first switch receives a network packet and the network packet containing address information that the forwarding database of the first switch has no corresponding address entry, or the network packet containing address information that a corresponding address entry in the forwarding database of the second network switch has expired.

In another aspect, the method also includes providing a plurality of first-level switches having a plurality of upward ports operating in a slave mode, providing a plurality of second-level switches including a channeling switch, operating at least one second-level switch operates in brain mode, operating least one second-level switch operates in a master mode, providing a plurality of local ports in the first-level switches for receiving and sending network packets, connecting the upward ports of the first-level switches to the second-level switches, sending the refresh packets to the second-level switches for synchronizing the forwarding databases of the second-level switches, and providing packet communications among the first-level switches, and providing refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

In still another aspect, the method additionally includes sending the refresh packet to each of the second-level switches except the channeling switch when one of the first-level switches sends a network packet to one of the second first-level switches through the channeling switch and needs to send a refresh packet containing the source address information of the network packet.

It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory and are intended to provide further explanation of the invention as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate the embodiments of the invention and, together with the description, serve to explain the objects, advantages, and principles of the invention.

In the drawings:

FIG. **1** is a functional block diagram of a network switch system with a known shared bus structure;

FIG. **2** is a functional block diagram of a network switch system with a known crossbar switching fabric structure;

FIG. **3** is a functional block diagram of a network switch system with a known ring-bus structure;

FIG. **4** is a functional block diagram of a network switch system with a known interleaved interconnection structure;

FIG. **5** is a functional block diagram of a network switch system with a known two-level structure;

FIG. **6** is a functional block diagram of a known non-blocking switch system;

FIG. **7** is a functional block diagram consistent with one embodiment of the present invention;

FIG. **8** is a block diagram of an exemplary lower-level port arrangement consistent with the present invention;

FIG. **9** is an exemplary address entry format in the forwarding database consistent with the present invention;

FIG. **10** is a flow chart showing the normal mode operation of a network switch consistent with the present invention;

FIG. **11** is a flow chart showing the slave mode operation of a network switch consistent with the present invention;

FIG. **12** is a flow chart showing the brain mode operation of a network switch consistent with the present invention; and

FIG. **13** is a flow chart showing the master mode operation of a network switch consistent with the present invention.

DESCRIPTION OF THE EMBODIMENTS

The present invention provides systems and methods for stacking network switches. The systems and methods synchronize forwarding databases of a network switch without requiring significant processing resources and provide the operational efficiency of a non-blocking system. The systems and methods may also employ multiple network switches of the same type to construct a network switch system with expanded number of ports.

In general, the systems and methods of the present invention operate in four different modes, normal, brain, master, and slave. A network switch is set to the normal mode of operation to enable a single network switch to operate independently. A plurality of network switches may also be combined and operate in different modes, including brain, master, and slave, for a stacked network switch system. The stacked configuration will provide more ports than a single network switch. In addition, the systems and methods for stacking network switches of the present invention are applicable to different types of network systems, including an Ethernet network system. In addition, the network switches may communicate with each other based on different types of network interfaces, such as MII, RMII and GMII interfaces in an Ethernet network environment.

FIG. **7** is a functional block diagram consistent with one embodiment of the present invention, specifically, an exemplary switch system that employs twelve eight-port switches to construct a thirty-two port switch system. Referring to FIG. **7**, the switch system includes a plurality of lower-level network switches L1–L8, and a plurality of upper-level switches U1 U4. The system sets lower-level network switches L1–L8, into the slave mode. The lower-level network switches L1–LB provide input/output ports to external devices, or specifically, to the users of the ports provided by each of the lower-level network switches L1–L8. The system sets one of the upper-level switches, for example, U1, to the brain mode (hereinafter "brain switch U1") and other upper-level switches U2 to U4 to the master mode (hereinafter "master switches U2–U4"). The brain switch U1 and master switches U2–U4 operate in part to channel network packets between different lower-level network switches. Each of the upper-level switches may connect with any of the lower-level switches to form a non-blocking switch system. As a result of the direct connection, all the upper-level switches U1–U4 become "neighboring switches" of each of the lower-level switches, and all of the lower-level switches become "neighboring switches" of each of the upper-level switches.

FIG. **8** is an exemplary port arrangement of any of the lower-level network switches shown in FIG. **7**. Referring to FIG. **8**, a lower-level network switch provides "upward ports" that connect with the upper-level network switches and "local ports" that provide input/output ports to the external devices or users of the lower-level network switch.

Taking an eight-port network switch as an example, the switch provides four upward ports and four local ports. The lower-level network switch sends refresh packets and network packets through the upward ports. In addition, the lower-level network switch treats the network packets as network trunk ports for ordinary network traffic.

The systems and methods of the present invention also provide refresh packets between the upper-level and lower-level switches to facilitate synchronization of forwarding databases in the system. A refresh packet is preferably a specifically identifiable packet containing source address information, such as the source address of a packet and the corresponding port information that needs to be registered by the system. When a switch in the system receives a refresh packet, the switch registers the source address information to the forwarding database of the switch. Preferably, the network switch system transmits the refresh packets within the system. Although the refresh packets might consume part of the system bandwidth, the systems and methods of the present invention provides a shorter Inter-Packet Gap ("IPG") or preamble to minimize bandwidth consumption.

Furthermore, a source-address learning delay may sometimes occur when the network system has not distributed refresh packets to all the switches. However, this delay only temporarily increases the broadcast traffic of the system and the broadcast traffic will diminish after the system has distributed the refresh packets to all of the switches. After synchronization, the forwarding databases of different switches have the same number of address entries. The contents of the address entries, however, may vary among the forwarding databases of different switches because the entries might contain different refresh timer values and port information for different neighboring switches.

FIG. **9** is an exemplary address entry format in the forwarding database consistent with the present invention. Referring to FIG. **9**, an entry of the forwarding database (not numbered) includes a refresh timer **16** that records the validity of a corresponding address entry in the forwarding databases of the neighboring switches within the system. As an example, the refresh timer **16** can be an up-counting timer or a down-counting timer that changes its value over time. When the counting value of the refresh timer **16** reaches a predetermined value, the switch recognizes that a corresponding address entry in the forwarding databases of the neighboring switches has "expired" and, therefore, needs to be updated. The switch resets the refresher timer **16** to its starting value either when the forwarding databases of the neighboring switches updates the corresponding entry or when the switch sends a refresh packet for that particular address entry.

Referring again to FIG. **9**, the entry of the forwarding database also includes an age timer **18**, address **20**, and port information **22**. The age timer **18** records the validity of the entry. As an example, the age timer **18** can be an up-counting timer or a down-counting timer that changes its value over time. When the counting value of the age timer **18** reaches a predetermined value, the switch recognizes that the entry in the forwarding database of the switch has "expired." The switch may then remove the expired entry from the forwarding database. The switch resets the age timer **18** to the starting value when the address entry is updated, usually through a source-address learning process, which may occur right after the switch receives an incoming packet or a refresh packet. The address **20** contains the address information of the entry, such as a Media Access Control ("MAC") address for an Ethernet network. The port infor-

nation **22** includes associated port information for the address **20**, such as a port number ("PN").

The network switch systems and method of the present invention also provide refresh packets to synchronize forwarding databases of the switches in the system. In general, the timing of sending a refresh packet is preferably packet-driven. Transmission of a refresh packet may occur upon receiving a network packet or an incoming refresh packet. In addition, a refresh packet may be sent by one of three ways.

A refresh packet may be sent when the switch receives a packet containing source address information but the forwarding database of the switch has no corresponding address entry. As an example, when the arrival of a packet triggers the source-address information learning ("SA learning") of a switch and the switch indicates that the forwarding database do not contain a corresponding entry for that source address, the switch sends a refresh packet containing the source address information of the incoming packet to register the information in the forwarding databases of neighboring switches in the system.

A refresh packet may also be sent when the switch receives a packet having expired source address information in a corresponding address entry in the neighboring switches. As an example, when the arrival of a packet triggers the SA learning of a switch and the switch indicates that the corresponding source address entry in the forwarding database of the neighboring switches has expired, the system sends a refresh packet containing the updated source address information of the incoming packet to register the updated information in the forwarding databases of the neighboring switches. The switch, for example, may identify such expiration based on the refresh timer **18** as illustrated in FIG. **9**.

Finally, a refresh packet may be sent when the switch is in the brain mode and receives a refresh packet from neighboring switches.

As described previously, the systems and methods of the present invention operate in one of four modes. FIG. **10** is a flow chart showing the normal mode operation of a network switch consistent with the present invention. Referring to FIG. **10**, the switch conducts packet verification, such as Cyclic Redundancy Check ("CRC"), at step **40** in order to filter out packets that have data or transmission errors. The verification step **40** ensures that all the packet information is correctly received. The switch proceeds with SA learning at step **42** to register the source address and source port information to the forwarding database of the switch.

The switch then performs destination address ("DA") information lookup at step **44** to obtain the destination port information based on the destination address of the incoming packet. If the switch cannot obtain the destination port information, the switch will broadcast the packet in order to have the packet delivered. The switch sends the packet to another network switch or another port within the same switch at step **46** by requesting and setting up a point-to-point connection with the destination port arranged. The switch may include two or more ports available as channels for transmitting the packet data. To achieve efficient transmission, the switch may choose one port from several available ports by checking a destination portmap that provides traffic or load information of the available ports.

FIG. **11** is a flow chart showing the slave mode operation of a network switch, such as the network switches L1 through L8 illustrated in FIG. **7**, consistent with the present invention. Referring to FIG. **11**, a slave switch conducts packet verification, such as Cyclic Redundancy Check ("CRC"), at step **50** in order to filter out packets that have data or transmission errors. The verification step **50** also ensures that all the packet information of an incoming network packet is correctly received. If the arrival packet is a refresh packet, the slave switch also ensures that the network switch receives the refresh packet correctly and marks the refresh packet at step **50**. In addition, refresh packets from upward ports will be received correctly, and refresh packets from local ports will be discarded. The slave switch then determines at step **52** (1) whether the packet is from local ports, (2) whether the forwarding database of the switch contains any corresponding entry to the source address information of the incoming packet, and (3) whether a corresponding address entry of the source address in the forwarding database of neighboring switches has expired. If the slave switch determines that the packet is from the local ports and the forwarding database of the slave switch contains no corresponding entry to the source address information of the packet, or the corresponding address entry in the neighboring switches has expired, the slave switch sends a refresh packet through the upward ports of the switch to upper-level switches at step **54**. Referring to FIG. **7**, a slave switch, for example, L1, may send a refresh packet to all the upper-level switches U1-U4.

To avoid providing the source address information more than once to an upper-level switch, the slave switch cannot provide the refresh packet to an intermediate channeling switch, if one exists. A channeling switch, preferably an upper-level switch, provides the channel with communicating packet data between two lower-level switches. Because the network packet that goes through the channeling switch contains the same source address information as the refresh packet, the channeling switch does not need a refresh packet to synchronize the forwarding database. As an example, referring to FIG. **7**, when the slave switch L1 sends the network packet to an intermediate channeling switch, for example, U2, and needs to send the refresh packet containing the corresponding address information of the network packet, the slave switch L1 only sends the refresh packet to the other three switches U1, U3, and U4, and skip sending the refresh packet to the channeling switch U2. When the slave switch does send the refresh packet, the switch may also reset the refresh timer of the corresponding entry within the forwarding database of the switch to reflect that the corresponding address entry in the neighboring switches has been updated.

Referring again to FIG. **11**, if the slave switch determines that the packet is not from the local ports and is a refresh packet or the slave switch has completed step **54**, the slave switch proceeds with source address information learning at step **56**. The slave switch registers the source address information of an incoming refresh packet or an incoming network packet to the forwarding database of the salve switch. The slave switch also resets the age timer to reflect that the switch has updated the corresponding address entry.

For an incoming network packet, the slave switch performs destination address information lookup ("DA lookup") at step **58** to obtain the destination port information corresponding to the destination address of the packet. At step **60**, the slave switch sends the packet to another network switch or port within the same switch by requesting and setting up a point-to-point connection with the destination port arranged. The slave switch may have two or more ports available as channels for transmitting packet data. To achieve efficient transmission, the switch may choose one port from several available ports by checking a destination portmap that provides the traffic or load information of the available ports.

FIG. **12** is a flow chart showing the brain mode operation of a network switch, such as the network switches U1 illustrated in FIG. **7**, consistent with the present invention. Referring to FIG. **12**, the brain switch U1 conducts packet verification, such as Cyclic Redundancy Check ("CRC"), at step **62** in order to filter out packets that have data or transmission errors. The verification step **62** also ensures that all the packet information of an incoming network packet is correctly received. If the arrival packet is a refresh packet, the brain switch also ensures that the network switch receives the refresh packet correctly and marks the refresh packet at step **62**.

The brain switch determines the following conditions at step **64** (1) whether the packet is a refresh packet, (2) whether the forwarding database of the brain switch contains any address entry corresponding to the source address information of the incoming network packet, and (3) whether the corresponding address entry of the source information in the forwarding databases of neighboring switches has expired. If the packet is a refresh packet, the brain switch sends the refresh packet to the neighboring switches. The brain switch may skip sending the refresh packet to an initiating switch. For example, referring to FIG. **7**, if the brain switch U1 receives a refresh packet from an initiating lower-level switch, for example, the switch L2, the brain switch U1 may skip sending the same refresh packet to the switch L2 because the switch L2 has already updated the same address information. When the brain switch U1 sends the refresh packet, the brain switch may also reset the refresh timer of the corresponding entry within the forwarding database to reflect that the corresponding address entry in the neighboring switches has been updated.

Referring again to FIG. **12**, if the packet is not a refresh packet, but the forwarding database of the brain switch contains no source address information of the network packet or the corresponding address entry in the neighboring switches has expired, the brain switch sends the refresh packet to neighboring switches within the system. The brain switch may skip sending the refresh packet to a source switch that sends the network packet to the brain switch. For example, referring to FIG. **7**, if the brain switch U1 receives a network packet from a source switch, for example, switch L3, the brain switch may skip sending the refresh packet to the source switch L3 because the source switch L3 has already updated the corresponding address information. Similarly, when the brain switch sends the refresh packet, the brain switch may also reset the refresh timer of the corresponding entry within the forwarding database to reflect that the corresponding address entry in the neighboring switches has been updated.

Referring to FIG. **12**, if none of the conditions of step **64** is met or the brain switch has completed step **66**, the brain switch proceeds with source address information learning at step **68**. The brain switch registers the source address information of an incoming refresh packet or an incoming network packet to the forwarding database of the switch. The brain switch also resets the age timer to reflect that the switch has updated the corresponding address entry.

For an incoming network packet, the brain switch performs destination address information lookup ("DA lookup") at step **70** to obtain the destination port information corresponding to the destination address of the packet. At step **72**, the brain switch sends the network packet to another network switch by requesting and setting up a point-to-point connection with the destination port arranged. The brain switch might have two or more ports available as channels for transmitting packet data. To achieve efficient transmission, the brain switch may choose one port from several available ports by checking a destination portmap that provides the traffic or load information of the available ports.

FIG. **13** is a flow chart showing the master mode operation of a network switch, such as the network switches U2, U3 or U4, illustrated in FIG. **7**, consistent with the present invention. The master switch conducts packet verification, such as Cyclic Redundancy Check ("CRC"), at step **74** to filter out packets that have data or transmission errors. The verification step **74** ensures that all the packet data of an incoming network packet is correctly received. If the arrival packet is a refresh packet, the master switch also ensures that the network switch receives the refresh packet correctly and marks the refresh packet. The master switch proceeds with source address information learning at step **76**. The master switch registers the source address information of an incoming refresh packet or an incoming network packet to the forwarding database of the master switch.

For an incoming network packet, the master switch performs destination address information lookup ("DA lookup") at step **78** to obtain the destination port information corresponding to the destination address of the packet. At step **80**, the master switch sends the packet to another network switch or another port within the same switch by requesting and setting up a point-to-point connection with the destination port. The master switch might have two or more ports available as channels for transmitting packet data. To achieve efficient transmission, the master switch may choose one port from several available ports by checking a destination portmap that provides the traffic or load information of the available ports.

As the above paragraphs illustrate, the present invention provides systems and methods for stacking network switches. The systems and methods provide combination of network switches and their operations in different modes. The systems and methods enable the same type of switches to be combined without requiring additional devices, such as external CPUs, switching fabrics, special bus or wiring arrangements, or new interfaces. The systems and methods of the present invention improve the operational efficiency of network switching system, provide improved bandwidth of switch systems, and offer a cost-effective approach to constructing or operating network systems with expanded number of ports.

It will be apparent to those skilled in the art that various modifications and variations can be made in the disclosed systems and methods without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

What is claimed is:

1. A network switch system, comprising:

a plurality of network switches for providing an exchange of network packets, each of the network switches comprising;

a forwarding database; and

a component providing at least one refresh packet, provided upon receiving the network packets, to synchronize the forwarding databases of the plurality of network switches;

wherein each of the plurality of network switches registers the at least one refresh packet to the forwarding database of the network switch upon receiving the refresh packets, and

wherein the plurality of network switches include a first switch and a second switch, each having a forwarding database, the first switch sending a refresh packet to the second switch when

a) the first switch receives a network packet containing address information indicating that the forwarding database of the first switch has no corresponding address entry, or

b) the first switch receives a network packet containing address information indicating that a corresponding address entry in the forwarding database of the second network switch has expired.

2. The system as claimed in claim 1, wherein the forwarding databases include at least one refresh timer in an address entry for recording the validity of a corresponding address entry in the forwarding databases of neighboring switches.

3. The system as claimed in claim 1, wherein the forwarding databases include an address entry having an age timer for the address entry that records the validity of the address entry, an address for the address entry, and associated port information for the address.

4. The system as claimed in claim 1, wherein the plurality of network switches further comprises,

a plurality of first-level switches having a plurality of upward ports operating in a slave mode,

a plurality of second-level switches including a channeling switch, wherein each of the first-level switches is configured to connect to each of the second-level switches,

wherein at least one second-level switch operates in a brain mode and at least one second-level switch operates in a master mode,

wherein the first-level switches providing a plurality of local ports for receiving and sending network packets, the upward ports of the first-level switches connecting to the second-level switches, the first-level switches sending the refresh packets to the second-level switches for synchronizing the forwarding databases of the second-level switches, and

wherein the second-level switches providing packet communications among the first-level switches, the second-level switch operating in the brain mode providing refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

5. The system as claimed in claim 4, wherein when one of the first-level switches sends a network packet to one of the second first-level switches through the channeling switch and needs to send a refresh packet containing the source address information of the network packet, the first-level switch sends the refresh packet to each of the second-level switches except the channeling switch.

6. The system as claimed in claim 4, wherein when a second-level switch needs to send a refresh packet from an initiating first-level switch, the second-level switch sends the refresh packet to each of the first-level switches except the initiating first-level switch.

7. The system as claimed in claim 4, wherein when the a second-level switch needs to send a refresh packet containing the source address information of a network packet from a first-level source switch, the second-level switch sends the refresh packet to each of the first-level switches except the first-level source switch.

8. The system as claimed in claim 4, wherein each of the first-level switches employs the upward ports as trunk ports for sending the network packets.

9. The system as claimed in claim 1, wherein the network switches are Ethernet switches.

10. A network switch system, comprising:

a plurality of first-level switches operating in a slave mode, the first-level switches providing a plurality of local ports for receiving and sending network packets; and

a plurality of second-level switches operating in one of brain mode or master mode,

wherein the first-level switches include a plurality of upward ports connecting to the second-level switches, each of the first-level switches and the second-level switches having a forwarding database,

wherein the first-level switches send the refresh packets to the second-level switches, for synchronizing the forwarding databases of the second-level switches, when

a) the first-level switch receives a network packet containing address information indicating that the forwarding database of the first-level switch has no corresponding address entry, or

b) the first-level switch receives a network racket containing address information indicating that a corresponding address entry in the forwarding databases of one of the second-level switches has expired,

wherein the second-level switches provide packet communications among the first-level switches, and

wherein a second-level switch operating in the brain mode provides refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

11. The system as claimed in claim 10, wherein each of the first-level and second-level switches registers the refresh packets information to the forwarding databases upon receiving the refresh packets.

12. The system as claimed in claim 10, wherein the forwarding databases of the first-level switches include at least one refresh timer in an address entry for recording the validity of a corresponding address entry in the forwarding databases of the second-level switches.

13. The system as claimed in claim 10, wherein the forwarding database of the second-level switch operating in the brain mode includes a refresh timer in an address entry for recording the validity of a corresponding address entry in the forwarding databases of the first-level switches.

14. The system as claimed in claim 10, wherein the forwarding databases of the first-level and second-level switches include an address entry containing an age timer for the address entry to record the validity of the address entry, an address for the address entry, and associated port information for the address.

15. The system as claimed in claim 10, wherein the second-level switch operating in the brain mode sends a refresh packet to the first-level switches connected with the second-level switch if the second-level switch operating in the brain mode receives a network packet, and the network packet contains address information that the forwarding database of the second-level switch operating in the brain mode has no corresponding address entry, the network packet containing address information that corresponding address entries in the forwarding databases of the first-level switches have expired, or the second-level switch operating in the brain mode receives an incoming refresh packet from the first-level switches.

16. The system as claimed in claim 10, wherein when one of the first-level switches sends a network packet to one of the second first-level switches through a channeling second-level switch and needs to send a refresh packet containing the source address information of the network packet, the first-level switch sends the refresh packet to each of the second-level switches except the channeling second-level switch.

17. The system as claimed in claim 10, wherein when one of the second-level switches needs to send a refresh packet coming from a first-level initiating switch or containing the source address information of a network packet from a first-level source switch, the second-level switch sends the refresh packet to each of the first-level switches except the first-level initiating switch and the first-level source switch.

18. A method for operating a network switch in a slave mode within a network switch system, the slave switch having a forwarding database, comprising:

receiving a network packet;

sending an outgoing refresh packet to neighboring switches when

a) the network packet contains address information indicating that the slave switch has no corresponding record in the forwarding database of the slave switch, or

b) the network packet contains address information indicating that corresponding address entries in the forwarding databases of the neighboring switches have expired, wherein the slave switch sends the outgoing refresh packet to synchronize the forwarding databases of the neighboring switches;

registering the address information of the network packet to the forwarding database of the slave switch; and

registering the address information of an incoming refresh packet to the forwarding database of the slave switch upon receiving the incoming refresh packet.

19. The method as claimed in claim 18, further comprising:

looking up the destination port of the network packet in the forwarding database of the slave switch;

sending the network packet to the destination port; and

broadcasting the network packet when the forwarding database of the slave switch has no corresponding destination port information for the network packet.

20. The method as claimed in claim 18, wherein the forwarding database of the slave switch includes a refresh timer in an address entry for recording the validity of a corresponding address entry in the forwarding databases of the neighboring switches.

21. The method as claimed in claim 18, wherein the forwarding database of the slave switch includes an address entry containing an age timer for the address entry that records the validity of the address entry, an address for the address entry, and associated port information for the address.

22. The method as claimed in claim 18, wherein the slave switch provides local ports for receiving and sending network packets and upward ports connecting to the neighboring switches.

23. The method as claimed in claim 18, wherein when the slave switch sends the network packet to a channeling switch and needs to send the outgoing refresh packet containing the source address information of the network packet, the slave switch sends the outgoing refresh packet to all the neighboring switches except the channeling switch.

24. The method as claimed in claim 18, wherein the neighboring switches includes network switches operating in a brain mode or a master mode.

25. A method of operating a network switch in a brain mode within a network switch system, the network switch having a forwarding database, comprising:

receiving a network packet;

sending an outgoing refresh packet to neighboring switches when

a) the network packet contains address information indicating that the forwarding database of the brain switch has no corresponding address entry or

b) the network packet contains address information indicating that corresponding address entries in the forwarding databases of the neighboring switches have expired; and

registering the address information of the network packet to the forwarding database of the brain switch,

wherein the brain switch receives an incoming refresh packet from the neighboring switches, and sends the outgoing refresh packet to synchronize the forwarding databases of the neighboring switches.

26. The method as claimed in claim 25, further comprising:

looking up the destination port of the network packet in the forwarding database of the brain switch;

sending the network packet to the destination port; and

broadcasting the network packet if the forwarding database of the brain switch has no corresponding destination port information for the network packet.

27. The method as claimed in claim 25, wherein the forwarding database of the brain switch includes a refresh timer in an address entry that records the validity of corresponding address entries of the forwarding databases of the neighboring switches.

28. A method of operating a network switch in a master mode within a network switch system, the master switch having a forwarding database, comprising:

receiving an incoming network packet;

registering address information of an incoming refresh packet to the forwarding database of the master switch upon receiving the incoming refresh packet;

looking up the destination port of the incoming network packet in the forwarding database of the master switch;

sending the incoming network packet to the destination port; and

broadcasting the incoming network packet when the forwarding database does not have corresponding destination port information,

wherein

a neighboring switch of the master switch sends the refresh packet to the master switch when

a) the neighboring switch receives an original network packet, and the original network packet contains address information indicating that the forwarding database of the neighboring switch has no corresponding address entry, or

b) the neighboring switch receives an original network packet, and the original network packet contains address information indicating that a corresponding address entry in the forwarding databases of the master switch has expired.

29. The method as claimed in claim 28, wherein the incoming refresh packet contains address and corresponding port information.

30. The method as claimed in claim 28, wherein the forwarding database of the master switch includes an

address entry containing an age timer for the address entry that records the validity of the address entry, an address for the address entry, and associated port information for the address.

31. The method as claimed in claim 28, wherein the master switch connects to neighboring switches within the network switch system and the neighboring switches operate in a slave mode.

32. A method for stacking network switches, comprising:

providing a plurality of network switches, including first and second switches, for providing an exchange of network packets;

providing a forwarding database for each of the network switches;

providing at least one refresh packet to synchronize the forwarding databases of the plurality of network switches upon receiving the network packets;

registering the refresh packet for each of the plurality of network switches to the forwarding databases

wherein the first switch sends a refresh packet to a second switch when

a) the first switch receives a network packet and the network packet contains address information indicating that the forwarding database of the first switch has no corresponding address entry, or

b) the first switch receives a network packet and the network packet contains address information that a corresponding address entry in the forwarding database of the second switch has expired.

33. The method as claimed in claim 32, further comprising,

providing a plurality of first-level switches having a plurality of upward ports operating in a slave mode,

providing a plurality of second-level switches including a channeling switch,

operating at least one second-level switch operates in a brain mode,

operating at least one second-level switch operates in a master mode,

providing a plurality of local ports in the first-level switches for receiving and sending network packets,

connecting the upward ports of the first-level switches to the second-level switches,

sending the refresh packets to the second-level switches for synchronizing the forwarding databases of the second-level switches, and

providing packet communications among the first-level switches, and providing refresh packets to the first-level switches for synchronizing the forwarding databases of the first-level switches.

34. The method as claimed in claim 33, further comprising sending the refresh packet to each of the second-level switches except the channeling switch when one of the first-level switches sends a network packet to one of the second first-level switches through the channeling switch and needs to send a refresh packet containing the source address information of the network packet.

35. The system as claimed in claim 33, further comprising sending the refresh packet to each of the first-level switches except the initiating first-level switch when one of the second-level switch needs to send a refresh packet from an initiating first-level switch.

36. The system as claimed in claim 33, further comprising sending the refresh packet to each of the first-level switches except the first-level source switch when one of the second-level switch needs to send a refresh packet containing the source address information of a network packet from a first-level source switch.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,139,267 B2
APPLICATION NO. : 10/087761
DATED : November 21, 2006
INVENTOR(S) : Kuo-Cheng Lu et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page, item (57), line 6, "includes" should read --include--.

Title page, item (57), line 10, "sends" should read --send--.

Claim 1, column 14, line 58, "comprising;" should read --comprising:--.

*Claim 7, column 15, line 62, "when the a" should read --when a--.

*Claim 10, column 16, line 24, "racket" should read --packet--.

Claim 24, column 18, line 2, "includes" should read --include--.

Claim 35, column 20, line 26, "switch" should read --switches--.

Claim 36, column 20, line 31, "switch" should read --switches--.

Signed and Sealed this

Twenty-seventh Day of March, 2007

JON W. DUDAS
Director of the United States Patent and Trademark Office

EXHIBIT B

US007236491B2

(12) **United States Patent**
Tsao et al.

(10) **Patent No.: US 7,236,491 B2**
(45) **Date of Patent: Jun. 26, 2007**

(54) **METHOD AND APPARATUS FOR SCHEDULING FOR PACKET-SWITCHED NETWORKS**

(75) Inventors: **Shih-Chiang Tsao**, Hsinchu (TW); **Ying-Dar Lin**, Hsinchu (TW); **Hui-Yang Huang**, Hsinchu (TW); **Chun-Yi Tsai**, Kaohsiung (TW)

(73) Assignee: **Industrial Technology Research Institute**, Hsinchu (TW)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 961 days.

(21) Appl. No.: **09/955,296**

(22) Filed: **Sep. 19, 2001**

(65) **Prior Publication Data**

US 2002/0131413 A1 Sep. 19, 2002

Related U.S. Application Data

(60) Provisional application No. 60/253,930, filed on Nov. 30, 2000.

(51) **Int. Cl.**
H04L 12/28 (2006.01)
H04L 12/56 (2006.01)

(52) **U.S. Cl.** **370/392**; 370/229; 370/395; 370/428

(58) **Field of Classification Search** 370/229-252, 370/322 392, 393 492, 412 428; 709/230 249
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,475,192	A *	10/1984	Fernow et al.	370/232
5,042,032	A *	8/1991	Dighe et al.	370/231
5,859,835	A *	1/1999	Varma et al.	370/229
6,047,000	A *	4/2000	Tsang et al.	370/412
6,134,217	A *	10/2000	Stiliadis et al.	370/232
6,359,861	B1 *	3/2002	Sui et al.	370/230
6,430,156	B1 *	8/2002	Park et al.	370/232
6,438,134	B1 *	8/2002	Chow et al.	370/412
6,470,016	B1 *	10/2002	Kalkunte et al.	370/395.41
6,532,213	B1 *	3/2003	Chiussi et al.	370/230.1

(Continued)

OTHER PUBLICATIONS

A. K. Parekh et al., "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Networking*, pp. 344-357, Jun. 1993.

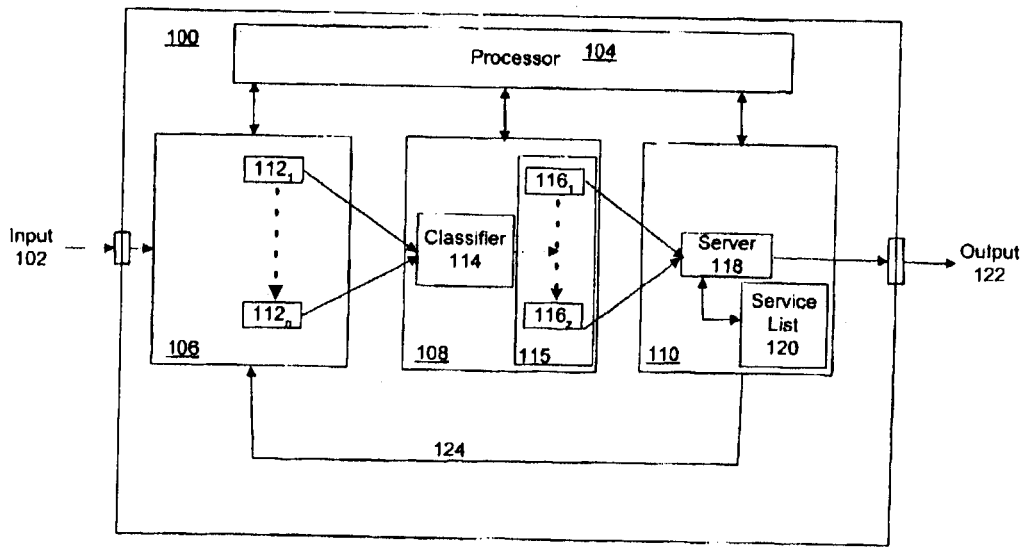
(Continued)

Primary Examiner—Man U. Phan
(74) *Attorney, Agent, or Firm*—Finnegan, Henderson, Farabow, Garrett & Dunner, L.L.P.

(57) **ABSTRACT**

Apparatus and method use pre-order queuing and scheduling. Reordering of the transmission sequence of packets that could be sent out in one round is allowed according to the quantum consumption status of the flow within the round. Per-packet time complexity is maintained independent from an increase in flow number and packets of variable-length are processed.

13 Claims, 7 Drawing Sheets



U.S. PATENT DOCUMENTS

6,532,501	B1 *	3/2003	McCracken	710/52
6,560,230	B1 *	5/2003	Li et al.	370/395.42
6,577,596	B1 *	6/2003	Olsson et al.	370/230
6,594,701	B1 *	7/2003	Forin	709/232
6,654,374	B1 *	11/2003	Fawaz et al.	370/394
6,674,721	B1 *	1/2004	Dittia et al.	370/235
6,714,517	B1 *	3/2004	Fawaz et al.	370/236
6,765,915	B1 *	7/2004	Metzger et al.	370/395.43
6,785,232	B1 *	8/2004	Kotser et al.	370/230.1
6,888,806	B1 *	5/2005	Miller et al.	370/316
6,891,835	B2 *	5/2005	Kalkunte et al.	370/395.41
6,920,109	B2 *	7/2005	Yazaki et al.	370/230.1
6,940,861	B2 *	9/2005	Liu et al.	370/395.21
2001/0043564	A1 *	11/2001	Bloch et al.	370/230
2002/0012348	A1 *	1/2002	Mizuhara et al.	370/392
2002/0071387	A1 *	6/2002	Horiguchi et al.	370/229
2002/0075875	A1 *	6/2002	Dravida et al.	370/395.21
2003/0133406	A1 *	7/2003	Fawaz et al.	370/229
2003/0189947	A1 *	10/2003	Beshai	370/428
2004/0228274	A1 *	11/2004	Yazaki et al.	370/229
2005/0163049	A1 *	7/2005	Yazaki et al.	370/230

OTHER PUBLICATIONS

J. C. R. Bennett et al., "WF^{2}Q: Worst-case fair weighted fair queueing," *Proc. IEEE INFOCOM '96*, pp. 120-128, San Francisco, CA, Mar. 1996.

S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proc. INFOCOM '94*, pp. 636-646, Jun. 1994.

L. Zhang, "Virtual Clock: A new traffic control algorithm for packet-switched networks," *ACM Trans. on Computer Systems*, vol. 9, No. 2, pp. 101-124, May 1991.

M. Shreedhar et al., "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Networking*, vol. 4, No. 3, pp. 375-385, Jun. 1996.

D. Stiliadis et al., "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Trans. Networking*, vol. 6, No. 2, pp. 175-185, Apr. 1998.

S. Suri, et al. "Leap forward virtual clock: a new fair queueing scheme with guaranteed delays and throughput fairness," *Proc. INFOCOM '97*, pp. 557-565, Apr. 1997.

D. Stiliadis et al., "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Networking*, vol. 6, No. 5, pp. 611-624, Oct. 1998.

N. Matsufuru et al. "Efficient fair queueing for ATM networks using uniform round robin," *Proc. INFOCOM '99*, pp. 389-397, Mar. 1999.

M. Katevenis et al., "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE Journal on Selected Areas in Communication*, vol. 9, No. 8, pp. 1265-1279, Oct. 1991.

H. M. Chaskar et al., "Fair scheduling with tunable latency: A Round Robin approach," *IEEE Globecom '99*, pp. 1328-1333, Dec. 1999.

J. C. R. Bennett et al., "High speed, scalable, and accurate implementation of packet fair queueing algorithms in ATM networks," *Proc. ICNP '97*, pp. 7-14, Oct. 1997.

V. Nageshwara Rao et al., "Concurrent accesss of priority queues," *Trans. on Computers*, vol. 37, No. 12, pp. 1657-1665, Dec. 1988.

J. L. Rexford et al., "Hardware-efficient fair queueing architectures for high-speed networks," *Proc. INFOCOM '96*, pp. 638-646, Mar. 1996.

* cited by examiner

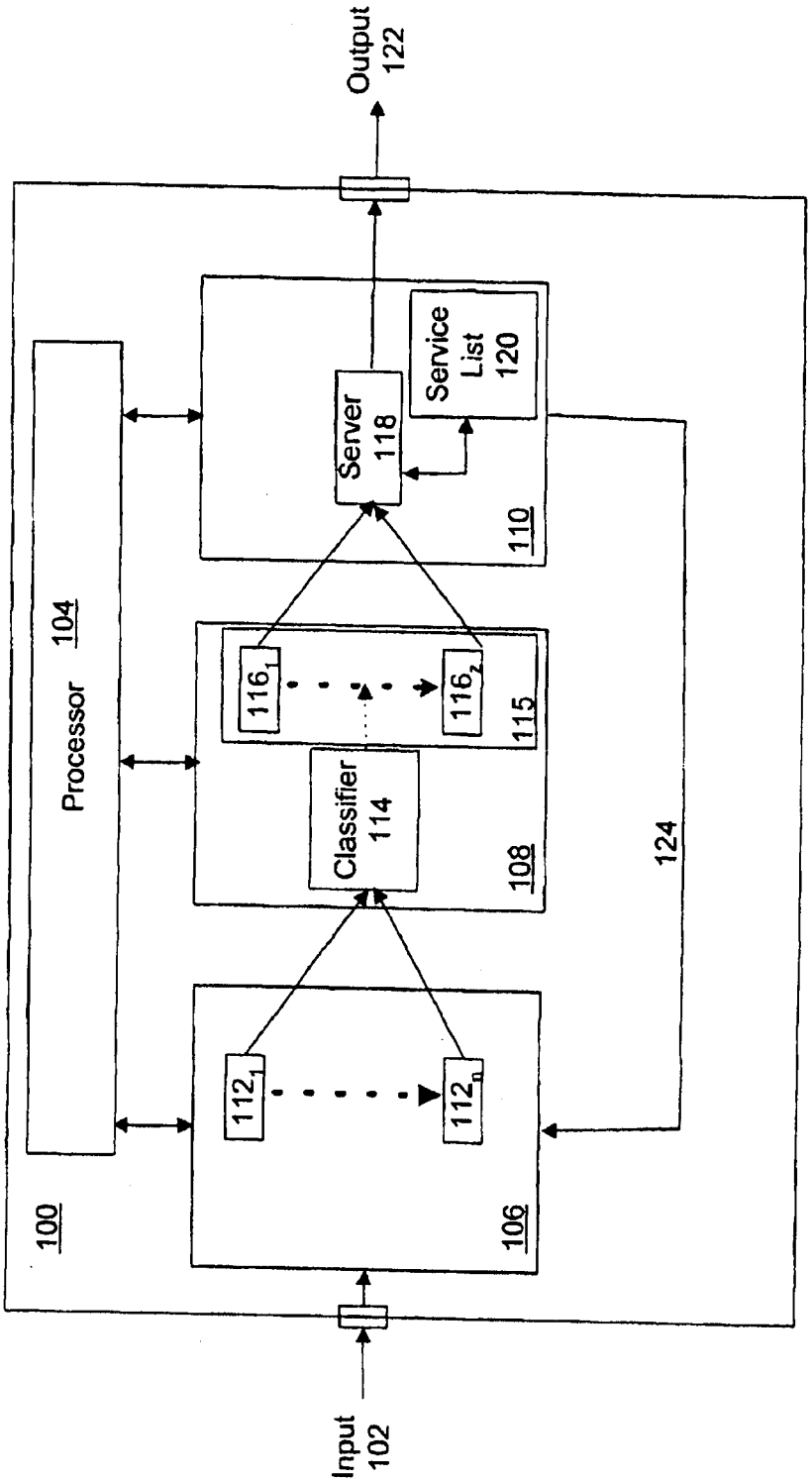

Output
122
Server
118
Service
List
120
110
Processor
104
116$_1$
116$_2$
115
Classifier
114
108
124
112$_1$
112$_n$
106
100
Input
102

Fig. 1

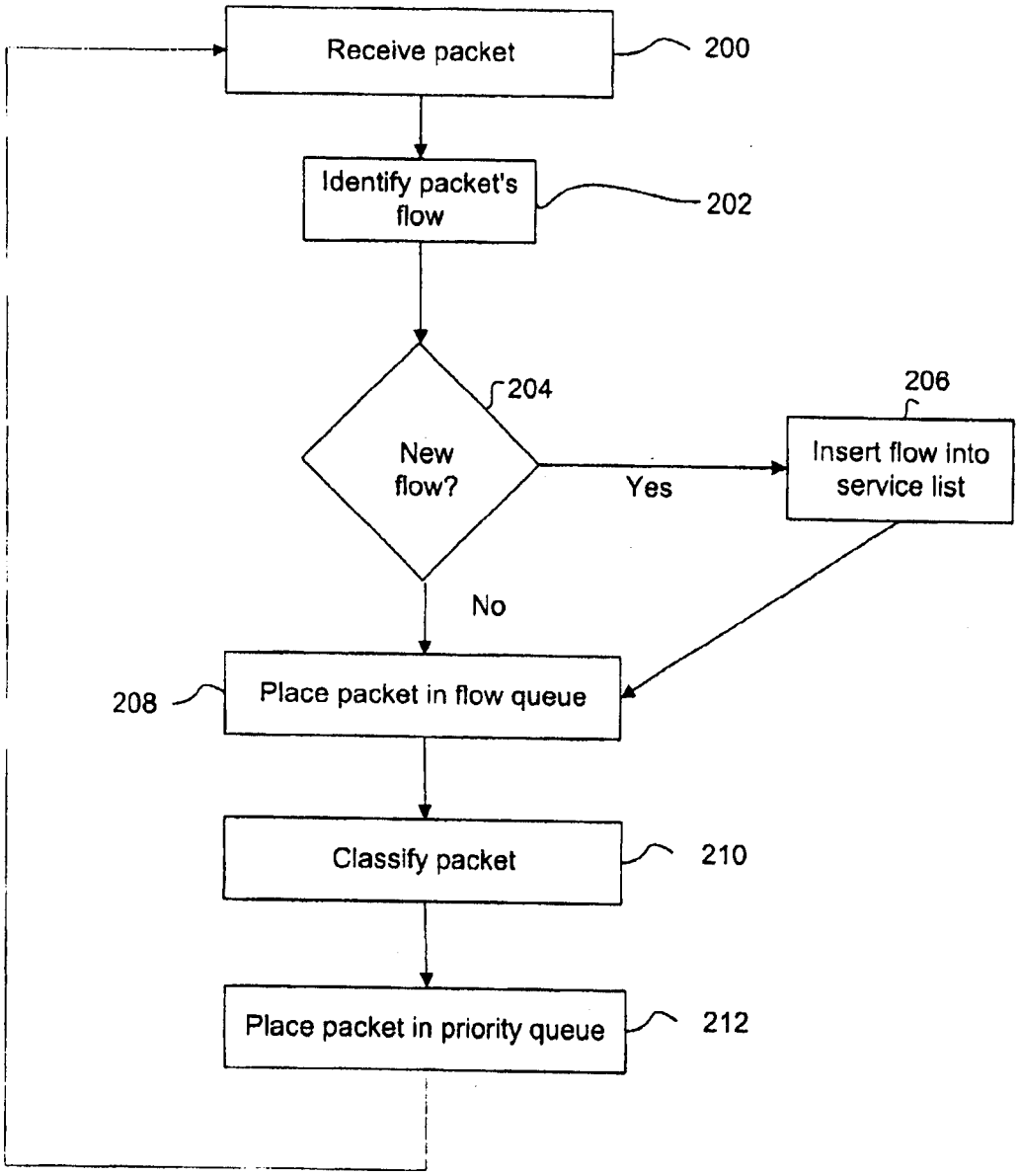

Receive packet
200
Identify packet's flow
202
204
New flow?
Yes
206
Insert flow into service list
No
208
Place packet in flow queue
Classify packet
210
Place packet in priority queue
212

Fig. 2

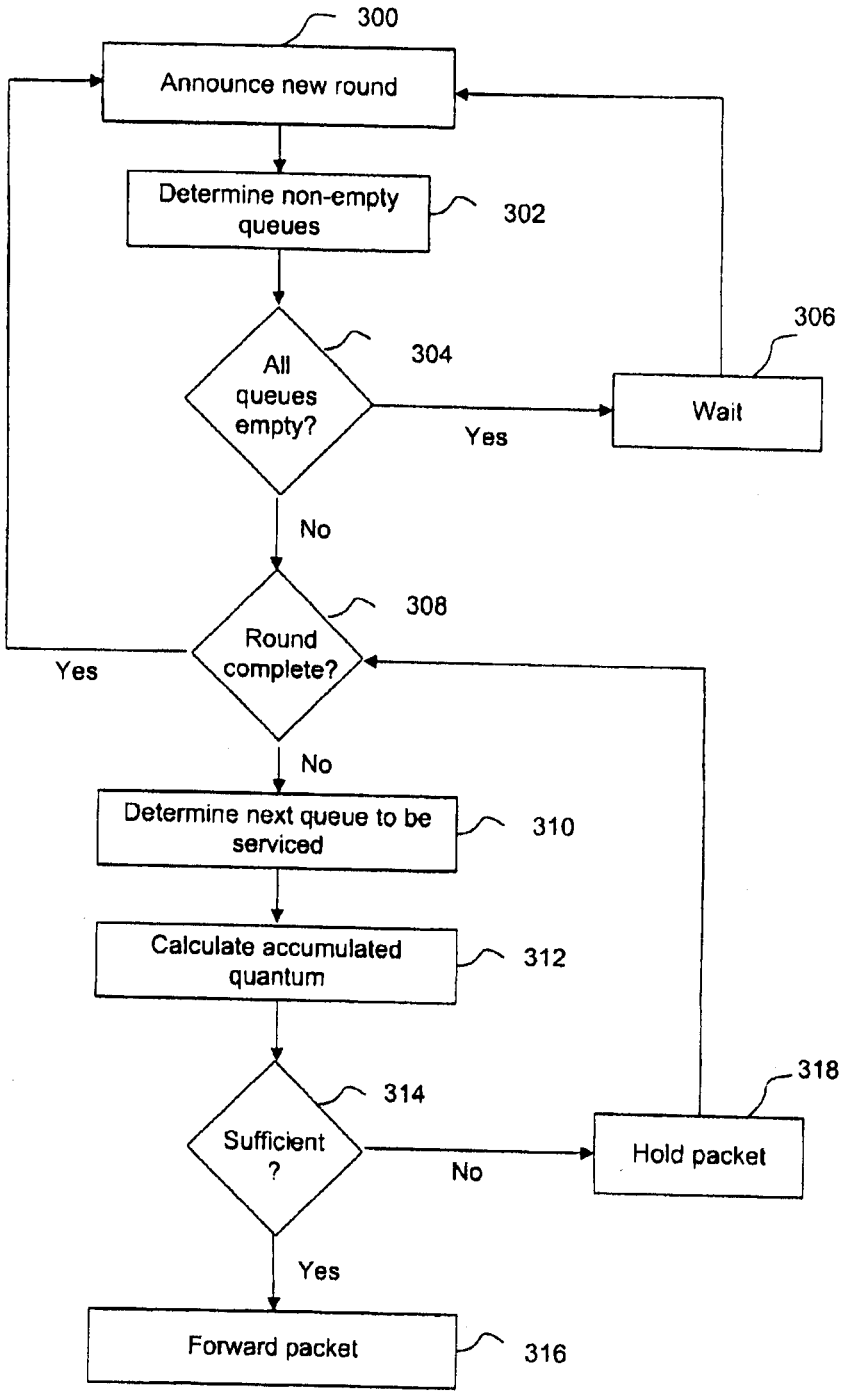

300
Announce new round
Determine non-empty queues
302
All queues empty?
304
Yes
306
Wait
No
Round complete?
308
Yes
No
Determine next queue to be serviced
310
Calculate accumulated quantum
312
Sufficient ?
314
No
318
Hold packet
Yes
Forward packet
316

Fig. 3

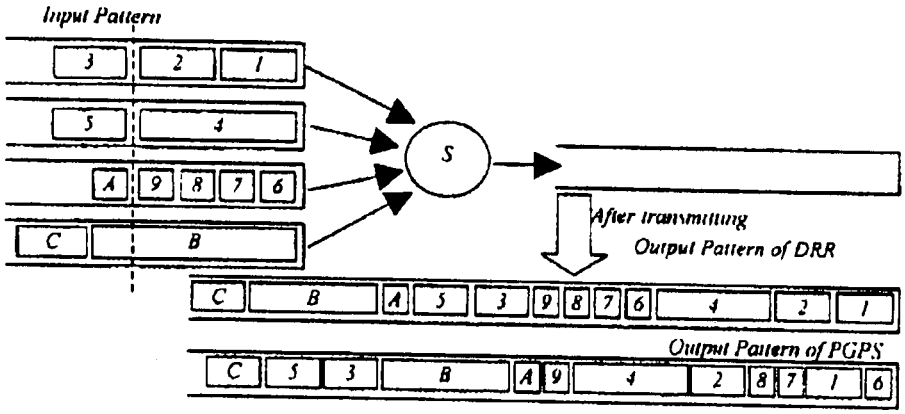

Input Pattern
S
After transmitting
Output Pattern of DRR
Output Pattern of PGPS

Fig. 4a

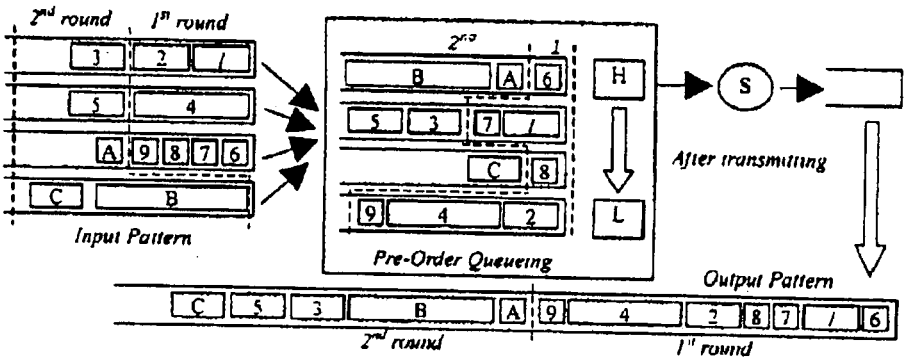

2nd round
1st round
Input Pattern
Pre-Order Queueing
H
L
S
After transmitting
Output Pattern
2nd round
1st round

Fig. 4b

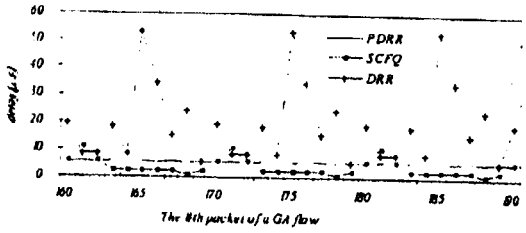

PDRR
SCFQ
DRR
160
165
170
175
180
185
190

Fig. 5

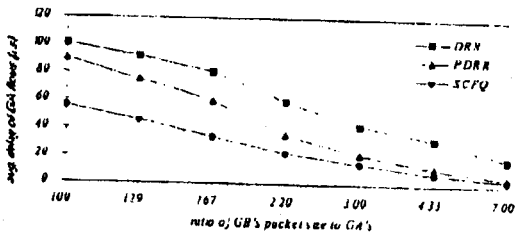

DRR
PDRR
SCFQ
100
129
167
220
300
433
700

Fig. 6

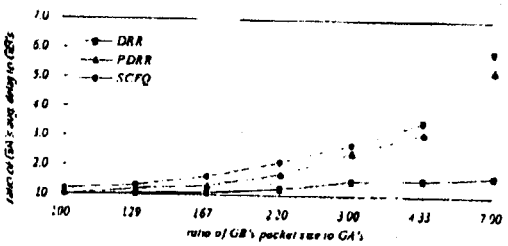

DRR
PDRR
SCFQ
7.0
6.0
5.0
4.0
3.0
2.0
1.0
1.00
1.29
1.67
2.20
3.00
4.33
7.90

Fig. 7

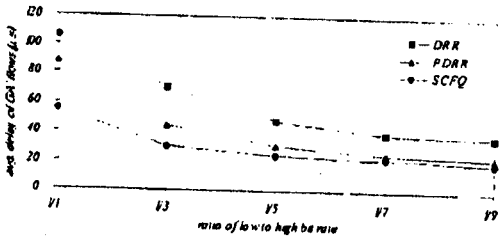

DRR
PDRR
SCFQ
120
100
80
60
40
20
0
V1
V3
V5
V7
V9

Fig. 8

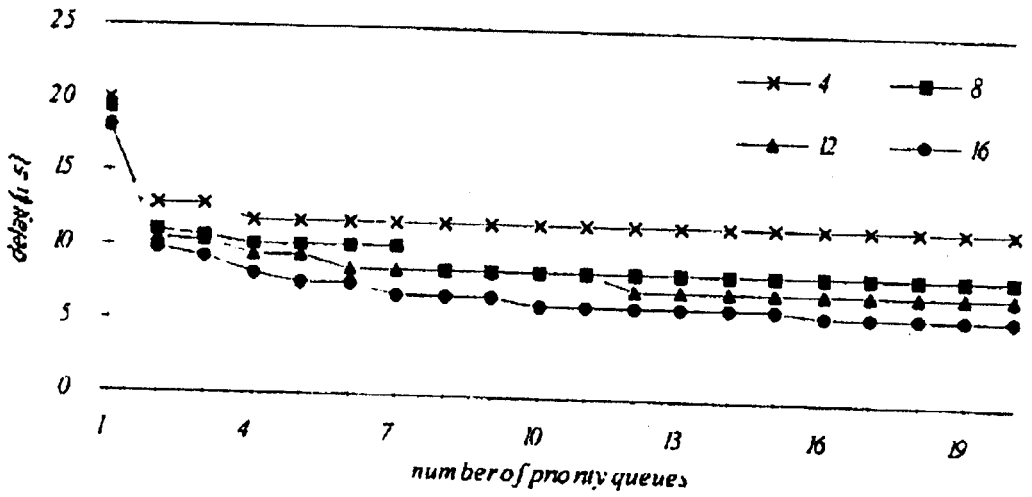

25
20
15
10
5
0
delay (μ s)
1
4
7
10
13
16
19
number of pronity queues
4
8
12
16

Fig. 9

METHOD AND APPARATUS FOR SCHEDULING FOR PACKET-SWITCHED NETWORKS

REFERENCE TO RELATED APPLICATIONS

This application claims priority from now abandoned prior provisional application Ser. No. 60/253,930, filed Nov. 30, 2000 for "PRE-ORDER DEFICIT ROUND ROBIN: A NEW SCHEDULING ALGORITHM FOR PACKET-SWITCHED NETWORKS."

FIELD OF THE INVENTION

This invention relates generally to packet scheduling. In particular, the invention relates to a packet scheduling method and apparatus for packet switched networks.

BACKGROUND OF THE INVENTION

In recent years, many packet scheduling algorithms have been proposed to reduce congestion, minimize delay (i.e., latency), and maintain fairness, especially to accommodate a high number of packet flows. Unfortunately, many of these algorithms can only be applied to fixed-size packets. Furthermore, many of these algorithms (even if they do not require fixed sized packets) exhibit poor performance as the number of packet flows increases.

Deficit Round Robin (DRR) is an algorithm which allows for variable-sized packets. Under DRR, a node rotationally selects packets to send out from all flows that have queued packets. During a round, each flow accumulates credits in discrete increments (e.g., in bytes) called a quantum. Unfortunately, DRR typically requires a quantum value that is very large, i.e., many times the size of the maximum packet size for a flow. The data presented in Table 1 below illustrates the above problem.

TABLE 1

THE TRAFFIC PARAMETERS AND QUANTUM SIZE OF 4 FLOWS

Flow ID	Reserved Rate (Mbps)	Traffic Type	Maximum Packet Size (byte)	Ratio of Max Packet Size to Reserved Rate	Quantum Size (byte)
A	128	CBR	400	250	512
B	16	CBR	640	320	640
C	64	CBR	800	100	2560
D	64	CBR	100	125	2560

The data in Table 1 assumes four flows, sharing the same link, and a link capacity of 160 megabits per second. As illustrated in Table 1, the quantum size of a flow can reach very large values relative to the flow's maximum packet size. For example, flow D has a quantum size of 25.6 times (2560/100) the maximum packet size. Unfortunately, due to the large quantum size required by DRR, DRR's performance can be poor compared to other algorithms such as Self-Clocked Fair Queuing (SCFQ) and Weighted Fair Queuing (WFQ).

However, simply reducing the quantum size also creates problems and is generally ineffective. For example, reducing the quantum size can cause a node using DRR to select no packets to send out after querying all flows in a particular round. This causes unacceptable delay and, again, causes poor performance. Thus, simply reducing the quantum sized used in a DRR node is generally not effective. Accordingly, it would be desirable to provide a scheduling algorithm and apparatus which does not require fixed size packets and exhibits good performance, including when there is a high number of packet flows.

SUMMARY OF THE INVENTION

In accordance with the invention, a method for scheduling a packet, comprises: receiving a packet; identifying a flow for the packet; classifying the packet based on the identified flow; and buffering the packet in one of a plurality of queues based on the classification of the packet.

In accordance with another aspect of the present invention, a system for scheduling a packet, comprises: an input to receive a plurality of packet; an arrival module to identify a flow for each of the plurality of packets; a classifier to assign each of the plurality of packets to one of a plurality of queues based on the identified flow; a server for selecting one of the plurality of queues based on a hierarchical order; and an output for outputting a packet from the selected queue.

In accordance with yet another aspect of the present invention, an apparatus for scheduling a packet, comprises: means for receiving a packet; means for identifying a flow for the packet; means for classifying the packet based on the identified flow; and means for buffering the packet in one of a plurality of queues based on the classification of the packet.

Additional advantages of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims.

It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments of the invention and together with the description, serve to explain the principles of the invention. In the drawings:

FIG. **1** illustrates a node **100** utilizing a Pre-Order Deficit Round Robin (PDRR) architecture in accordance with principles of the present invention;

FIG. **2** shows a method for scheduling packets in accordance with principles of the present invention;

FIG. **3** shows a method of transmitting packets in accordance with principles of the present invention;

FIG. **4***a* illustrates the operation of Deficit Round Robin (DRR) in comparison with weighted fair queuing (WFQ);

FIG. **4***b* shows the present invention using PDRR operating on the same input pattern and assumptions used in FIG. **4***a*;

FIGS. **5-8** show various simulation results to compare the performance of embodiments consistent with the present invention using PDRR with DRR and SCFQ; and

FIG. **9** shows performance consistent with the present invention as the number of priority queues is varied for a specific traffic environment.

DESCRIPTION OF THE EMBODIMENTS

Reference will now be made in detail to exemplary embodiments of the invention, examples of which are illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

Embodiments consistent with the present invention provide pre-order deficit round robin (PDRR) architecture to execute a scheduling algorithm which minimizes delay

while maintaining fairness in a packet switched network. Embodiments consistent with the present invention use O(1) per-packet time complexity in most cases (i.e., as the number of packet flows increases) and is amenable to variable-length packets.

Analysis results from testing embodiments consistent with the present invention with respect to three measures, including latency, fairness and per-packet time complexity are also provided. The analysis results provide supporting evidence that embodiments consistent with the present invention offer better performance in latency, fairness and lower time complexity. Furthermore, simulation results are provided to demonstrate the behavior of embodiments consistent with the present invention.

FIG. **1** illustrates a node **100** utilizing the PDRR architecture in accordance with principles of the present invention. In particular, node **100** comprises an input port **102**, a processor **104**, a packet arrival module **106**, a pre-order queuing module **108**, a packet departure module **110**, and an output port **122**.

Input port **102** interfaces node **100** to a link, e.g., to other nodes (not shown) and receives incoming packets. For purposes of illustration, node **100** is shown with one input port, i.e., input port **102**. However, node **100** may be implemented with any number of input ports for receiving incoming packets.

Processor **104** performs various operations for receiving, scheduling, and passing packets. Processor **104** may be implemented using hardware logic in combination with software and an operating system. Examples of the operating system and software include the UNIX operating system and the LINUX operating system for executing code written using C and C++.

Packet arrival module **106** receives packets from input port **102**, identifies each packet's flow, and places each packet in its corresponding flow queue. Packet arrival module **106** determines the number n and identification of flow queues 112_1-112_n based upon information received from packet departure module **110** via path **124**. Packet arrival module **106** may also provide notification, e.g., to packet departure module **110** via processor **104**, when a packet arrives for a new flow to be serviced by node **100**. Furthermore, packet arrival module **106** may notify pre-order queuing module **108**, e.g., if there are no other packets for a particular flow. As shown in FIG. **1**, packet arrival module **106** comprises a set of flow queues **112**-112_n, where n is the number of flows currently being serviced by node **100**. Packet arrival module **106** may be implemented using any combination of hardware logic and software. Packet arrival module **106** may use processing functions of processor **104** to execute instructions in software. Below is one example of pseudo-code called "PKT_Arrival" which may be used by packet arrival module **106** to place each packet into its corresponding flow Fq (i.e., one of flow queues 112_1-112_n).

PKT_Arrival module

```
i←ExtractFlow(p)              // Get the flow # of packet p
Enqueue(p, Fq_i)
If NumItem(Fq_i)=1 Then       // The equal implies that Fq_i is
    SendMsg(PKT_Pass, i)      // empty before p was placed into it,
                              // and need to notify PACKET_Pass to
                              // handle the flow i
```

Output port **122** outputs packets passed from node **100** on to a link, e.g., to other nodes. For purposes of illustration, node **100** is shown with one output port, i.e., output port **122**. However, node **100** may be implemented with any number of output ports. In addition, node **100** may be implemented with dual purpose ports which function as both an input and output port.

Pre-order queuing module **108** places packets from non-empty flow queues **112**-**112***n* into a second set of queues. Pre-order queuing module **108** may process any size packet. Pre-order queuing module **108** comprises a classifier sub-module **114** and a priority queuing sub-module **115**.

Classifier sub-module **114** retrieves packets from each non-empty flow queue, i.e., flow queues 112_1-112_n within packet arrival module **106**, determines a priority for each packet, and places each packet in an appropriate priority queue, i.e., priority queues 116_1-116_z, within priority queuing sub-module **115**. In addition, classifier sub-module **114** may enable a packet (e.g., for a high priority flow) to be considered for transmission immediately. Priority queuing sub-module **115** maintains a number of priority queues 116_1-116_z, where z is represents the number of priorities used by classifier sub-module **114**.

Pre-order queuing module **108** may implement classifier sub-module **114** and priority queuing sub-module **115** in a wide variety of ways. For example, pre-order queuing module **108** may use processing functions of processor **104** and execute instructions in software. Below is one example of pseudo-code called "PKT_Pass" which may be used by pre-order queuing module **108**. PKT_Pass decides to which class j a packet belongs and places the packet into the corresponding priority queue Pq_j (i.e., priority queues 116_1-116_z) from its Fq (i.e., flow queues 112_1-112_n).

PKT_Pass module

```
While(TRUE)
{i←WaitMsg( )                 // Wait until a packet is placed to the
                              empty Fq_i
 If Round_i≠Round_sys         // Non-equivalent implies a new
                              round is arrival
 { Round_i←Round_sys
   DC_i←Max(DC_i, Quantum_i)
 }
 While DC_i>0 and             // Classify eligible packets into PQ
 NonEmpty(Fq_i)
 { PktSize←Size(Head(Fq_i))   // Get the size of the packet at head of Fq_i
   If (PktSize<DC_i) Then     // if flow i credits are enough to
                              send out packets.
   { DC_i←DC - PktSize        // Take out the used credits
     j←Z - (DC_i Pqg_i)       // Compute the j
     Enqueue(Dequeue(Fq_i),   // Move the head packet of Fq_i to
     Pq_j)                    eligible Pq_j
     If NumItem(Pq_j)=1       //It implies that Pq_j is empty and j
                              is non-existed in
     Then                     // the min heap before the last packt
                              was placed
                              into it
        MH  Insert(j)         //Insert j to min heap
   }
 }
 If NonEmpty(Fq_i) Then       // Imply the residual credits are not enough
 { Enqueue(i, AckList)
   If NumItem(AckList)=1 Then SetEvent(EV_server)
 }
}
```

Packet departure module **110** retrieves packets from non-empty priority queues, i.e., priority queues 116_1-116_z, within priority queuing sub-module **115** and outputs packets to output port **122**. Packet departure module **110** comprises a server sub-module **118** and a service list **120**.

Server sub-module **118** services in a round-robin fashion each non-empty priority queue among priority queues 116_1-116_z. Server sub-module **118** refers to service list **120** to

determine the non-empty priority queues. In one embodiment, server sub-module **118** declares a new "round" of service and services the non-empty priority queues among priority queues **116**$_1$-**116**$_z$ using, e.g., an algorithm similar to a deficit round robin (DRR) algorithm. Server sub-module **118** in conjunction with service list **120** may use a quantum and a deficit counter for each flow of packets to determine how a particular priority queue is serviced. The quantum represents a share of available bandwidth (e.g., in bytes) allocated to a flow within the period of one round. A quantum$_i$ for a flow i can be expressed as

$$Quantum_i = \frac{r_i}{C} \times F, \tag{1}$$

where r_i is the rate allocated to flow i, C is the link service rate, and F is the frame size that represents the summation of quantums for all flows.

A flow accumulates shares of bandwidth (i.e., in quantum increments) during a round. The deficit counter accumulates any residual quantum of flow i in the (j−1)th round, which can be represented as DeficitCounter$_i^{j-1}$. The next time that flow i is serviced by a node, an additional DeficitCounter$_i^{j-1}$ bytes of data (i.e., incremented by quantum$_i$) can be sent out in the jth round. Server sub-module **118** verifies the size of the packet at the head of the priority queue ("head packet") currently being serviced. As described with reference to FIG. **3** below, server sub-module **118** also determines when a particular packet will be transmitted, e.g., via output port **122**.

Server sub-module **118** maintains service list **120**. As noted above, service list **120** includes data for all flows currently being serviced by node **100**. For each flow, service list **120** may include a flow identifier, a deficit counter, and a quantum. If a flow has no packets, e.g., within flow queues **112**$_1$-**112**$_n$, server sub-module **118** may delete the flow identifier from service list **120**. Alternatively, when a packet arrives for a new flow, server sub-module **118** may add an identifier for the new flow to service list **120**.

In one embodiment, server sub-module **118** updates the deficit counters (DeficitCounter$_i^j$) in service list **120** according to the equation:

$$\text{DeficitCounter}_i^j = \text{DeficitCounter}_i^{j-1} + \text{Quantum}_i.$$

As noted above, the quantum indicates the increments of shares of bandwidth accumulated by a particular flow. Server sub-module **118** calculates the quantum such that a packet can be processed in O(1) operations. The quantum for a flow may be larger than the maximum packet size within the flow so that at least one packet per backlogged flow can be served in a round. The quantum for any two flows i and j may be expressed by

$$\frac{Quantum_i}{Quantum_j} = \frac{r_i}{r_j}. \tag{2}$$

Even assuming that all flows begin heavily backlogged at time t, the principles of the present invention allow server sub-module **118** to exhibit good performance. That is, server sub-module **118** can send out the packet with the earliest virtual finishing timestamp first among the packets at the heads of all flow queues. Under the above heavy backlog assumption, the virtual finishing timestamp of a packet may be computed as

$$TS_i^m = TS_i^{m-1} + \frac{L_i^m}{r_i}. \tag{3}$$

where TS_i^m denotes the timestamp of the mth packet of flow i after time t and, for all i, TS_i^0 is set to zero at time t, r_i denotes the allocated rate of flow i, and L_i^m denotes the size of the mth packet of flow i after time t. Equation (3), by substituting Acc_i^m for $TS_i^m \times r_i$, is equivalent to

$$\frac{Acc_i^m}{Quantum_i} = \frac{Acc_i^{m-1} + L_i^m}{Quantum_i}. \tag{4}$$

where Acc_i^m denotes the accumulated amount of data within a byte that flow i has sent out after transmitting the mth packet after time t. Assume that all m packets could be transmitted in the kth round. Equation (4), by replacing Acc^m with $\text{DeficitCounter}_i^0 - \text{DeficitCounter}_i^m$, is equivalent to

$$\frac{DeficitCounter_i^m}{Quantum_i} = \frac{DeficitCounter_i^{m-1} - L_i^m}{Quantum_i}. \tag{5}$$

where $\text{DeficitCounter}_i^m$ denotes the residual quantum of flow i in this round after it puts the mth packet into the Pre-order Queuing. To further illustrate the equivalence, the following definition is provided:

Definition 1: The Quantum Availability, QA_i^m, of the packet P_i^m is the ratio of its $\text{DeficitCounter}_i^m$ to Quantum_i, i.e.

$$QA_i^m = \frac{DeficitConter_i^m}{Quantum_i}. \tag{6}$$

Lemma 1: For any packet P_i^m, its Quantum Availability QA_i^m satisfies

$$0 \le QA_i^m < 1.$$

Lemma 2: For the packet with the smallest timestamp in one round, its QA is the largest.

Accordingly, the server sub-module **118** selects the packet with the largest QA within one round to send out. However, to avoid having to search for the packet with the largest QA among all packets that could be sent out in a round, classifier sub-module **114** classifies packets into several classes according to their QA and places them into the corresponding priority queues, i.e., priority queues **116**$_1$-**116**$_z$.

There are Z priority queues and, hence, Z classes. For the mth packet of flow i that can be sent out in this round, its class n_i^m can be derived as

$$n_i^m = Z - \lfloor QA_i^m \times Z \rfloor = Z - \left\lfloor \frac{DeficitCounter_i^m}{Pqg_i} \right\rfloor, \tag{7}$$

where $\text{DeficitCounter}_i^m$ denotes the residual credits in byte for flow i in the kth round after the mth packet is placed into a priority queue, and Pqg_i denotes the granularity of priority queue for flow i derived as

$$Pqg_i = \frac{Quantum_i}{Z} \quad (8)$$

Below is one example of pseudo-code called "PKT_Departure" which may be used by packet departure module **110**.

```
PKT_Departure module
While(TRUE)
{ If MH_Empty( ) Then                    // Imply no packets can be placed
                                         into PQ
    { Round_sys←Round_sys+1              // Announce arrival of the new round
      EC←WaitEvents(EV_minheap, EV_system) // wait until a packet was placed in PQ or
                                         // any Fq's
      If ( EC = EV_actlist)              // Imply there are packets without sending
                                         out at last
      Then                               round
      { NumAckList←NumItem(AckList)      // There are NumAckList
                                         non-empty
                                         // Fq's at last round
        While(NumAckList>0)              //For non-empty Fq's at
                                         last round, accumulate
        { i←Dequeue(AckList)             // their residual credits for
          DC_i←DC_i + Quantum_i          using at this round
          SendMsg(PKT_Pass, i)           // Ack PKT_Pass to pass
                                         packets of Fq_i into PQ.
          NumAckList←NumAckList-1
        }
        WaitEvent(EV_minheap)            // Pause if no packets in PQ
      }   // IF EC=EV_actlist
}       // IF MH_EMPTY
WaitEvent(ServerIdle)                    // Pause if the server is
                                         sending out a packet
MH_Lock( ) //To avoid the MinHeapRoot being modified as
           MH_Delete( ) is involved.
If Empty(Pq_MinHeapRoot) Then MH_Delete( )
MH_Unlock( )
Send(Dequeue(Pq_MinHeapRoot))            // Send out the packet in
                                         the Pq_i with the smallest i
                                         // among non-empty Pq_i
```

Tables 2 and 3 provide some definitions used by the pseudo-code modules described above.

TABLE 2

THE OPERATIONS USED IN PKT_ARRIVAL AND PKT_DEPARTURE AND PKT_PASS PROCEDURES

Operation	Description
Enqueue, Dequeue, NonEmpty, Empty, Head	The standard Queue operations
NumItem(A)	Return the number of entries in the A
MH_Insert(x), MH_Delete(), MH_Empty()	The standard operations of the min heap
MinHeapRoot	The minimum value among nodes of the min heap
MH_Lock, MH_Unlock	After locking, only one module can access the min heap
SetEvent(ev)	Signal the event ev The event will remain until signaled someone releases it
EventCase = WaitEvents(ev1, ev2, .)	Once any event is in the signaled state, return it to EventCase and release it Note ev1 is prior to ev2
SendMsg(x,y)	Send message along with value y to x
y = WaitMsg()	Wait message and store the received value in y

TABLE 3

THE VARIABLES USED IN PKT_ARRIVAL AND PKT_DEPARTURE AND PKT_PASS PROCEDURES

Variable	Description
Fq_i	The queue of flow i, i = 1..N
Pq_j	Priority queue j, j = 1..Z
$Quantum_i$	The created allocated to flow i in one round
DC_i	The DeficitCounter of that flow i
Pqg_i	The granularity of priority queue for flow i
Z	The number of priority queues
$Round_{sys}$	The identification of the system current round
$Round_i$	The identification of the most of flow i
$EV_{minheap}$	A event signaled as one key was placed into the empty min heap
$EV_{actlist}$	A event signaled as one item was placed into the empty ActList

FIG. **2** shows a method for scheduling packets in accordance with the principles of the present invention. In step **200**, input port **102** receives a packet and provides the packet to packet arrival module **106**. In step **202**, packet arrival module **106** identifies the packet's flow. In step **204**, packet arrival module **106** determines whether the packet's flow is a new flow or a pre-existing flow. For example, packet arrival module **106** may work in conjunction with processor **104** to check service list **120**. If the flow is not found within service list **120**, e.g., by processor **104**, then processing flows to step **206** in which packet arrival module **106** sends a message to packet departure module **110** to add the new flow to service list **120**. Packet arrival module **104** may then add a flow queue to flow queues 112_1-112_n for the new flow. Processing then proceeds to step **208** as described below.

If the flow is found within service list **120**, then processing flow to step **208**. In step **208**, packet arrival module **106** places the packet in its corresponding flow queue, i.e., one of flow queues 112_1-112_n and sends a message to pre-order queuing module **108**.

In step **210**, upon receiving the message, pre-order queuing module **108** notifies classifier sub-module **114** to retrieve and classify the packet from flow queues 112_1-112_n and classifies the packet. Classifier sub-module **114** may classify the packet (and its flow) in a wide variety of ways. For example, classifier sub-module **114** may classify the packet based upon: a source address, a destination address, or other information such as a service class (e.g., constant bit rate), transport control protocol port, etc. Other information may also be used by classifier sub-module **114** to classify the packet.

In step **212**, classifier sub-module **114** places the packet in priority queues 116_1-116_z within priority queuing sub-module **115** based upon the packet's classification. Upon the placement of the packet in priority queues 116_1-116_z, pre-order queuing module **108** sends a message to packet departure module **110**. The message may include the size of the packet and the priority queue, i.e., one of priority queues 116_1-116_z, into which the packet was placed. Processing then returns to step **200**, where the process for scheduling packets repeats.

FIG. **3** shows a method of transmitting packets in accordance with the principles of the present invention. In particular, in step **300**, server sub-module **118** sends a message announcing a new round, e.g., to pre-order queuing module **108** via processor **104**. Server sub-module **118** may announce a new round at various times, e.g., upon packet

departure module **110** receiving a message that there are packets in priority queuing sub-module **115** and/or at pre-determined intervals.

In step **302**, server sub-module **118** determines which priority queues among priority queues **116**$_1$-**116**$_z$ are non-empty (i.e., have packets placed within them). Server sub-module **118** may determine the non-empty queues by searching service list **120**.

In step **304**, server sub-module **118** confirms that there are non-empty priority queues. If all priority queues **116**$_1$-**116**$_z$ are empty, then processing proceeds to step **306**. Otherwise, processing proceeds to step **308**. In step **306**, server sub-module **118** waits to announce a new round. Server sub-module **118** may wait to receive a message from pre-order queuing module **108**. Alternatively, server sub-module **118** may wait for a predetermined period of time. However, any type of wait period and/or initiation of a new round is within the principles of the present invention.

In step **308**, server sub-module **118** determines whether the current round is complete. For example, a round may be complete upon server sub-module **118** servicing all non-empty priority queues. If the round is complete, then processing flows back to step **300** where a new round is announced.

If the round is not complete, then processing flows to step **310** in which the server sub-module **118** determines a priority queue to service next. Server sub-module **118** may determine the next priority queue to service such that the priority queue among priority queues **116**$_1$-**116**$_z$ with the highest priority is serviced before serving lower priority queues. Alternatively, server sub-module **118** may service priority queues **116**$_1$-**116**$_z$ in a rotating fashion.

In one embodiment, a complete binary tree (called min_heap in the pseudo code), is used to enable server sub-module **118** to efficiently determine which non-empty priority queue among priority queues **116**$_1$-**116**$_z$ has the highest priority. For example, referring now to the pseudo code PKT_Pass and PKT_Departure described above, once packets are placed into priority queuing sub-module **115** (e.g., by PKT_Pass) the status of the event $EV_{minheap}$ is signaled to notify packet departure module **110** (e.g., PKT_Departure) to send out a packet. After transmission is complete, ServerIdle is signaled and PKT_Departure repeats the last action until the $Pq_{MinHeapRoot}$ is empty. The function MH_Delete may then delete the root node of the binary tree (i.e., min_heap) and set MinHeapRoot to the smallest j among residual nodes. When the min_heap is empty, i.e., all Pq_j's are empty, PKT_Departure may declare the arrival of a new round by adding 1 to $Round_{sys}$. For all non-empty Fq's, i.e., flows with packets remaining in the last round, PKT_Departure may update DeficitCounters according to the sequence in the service list (e.g., AckList) and request that PKT_Pass classify the eligible packets into the priority queuing sub-module **115**.

In step **312**, server sub-module **118** calculates the accumulated quantum (i.e., as indicated by DeficitCounter) for the packet. In step **314**, server sub-module **118** then determines whether the packet will be sent out during the current round. For example, If the packet's size is smaller than DeficitCounter$_i^j$, server sub-module **118** decreases DeficitCounter^j by the packet size and processing proceeds to step **316** where server sub-module **118** sends the packet out, e.g., via output port **122**. Alternatively, server sub-module **118** may reset DeficitCounter$_i^j$ to zero, i.e., the residual quantum remaining from the previous round cannot be carried over to avoid delaying service to other priority queues. In addition, during progress of a particular round, if a packet arrives in a priority queue having a higher priority than the priority queue currently being serviced, then server sub-module **118** may service the higher priority queue out of order within the round and send the packet out before any other packets from lower priority queues are sent out.

If the packet's size is larger than DeficitCounter$_i^j$, then processing proceeds to step **318** where server sub-module **118** may hold the packet until a subsequent round. Server sub-module **118** may repeatedly hold the packet until the size of the head packet is larger than DeficitCounter$_i^j$, i.e., there is insufficient residual quantum to serve a subsequent packet, or there are no remaining packets in the priority queue. During a subsequent round, the next time the priority queue gets its turn, server sub-module **118** may send out additional DeficitCounter$_i^j$ bytes of data in addition to quantum$_i$ bytes.

FIG. **4***a* illustrates the operation of Deficit Round Robin (DRR) in comparison with weighted fair queuing (WFQ). As shown in FIG. **4***a*, there are four flows requesting the same amount of bandwidth and having fixed, but heterogeneous packet sizes. The same quantum is assigned to all of them and, according to the known DRR algorithms, the quantum should be equal to the largest maximum packet size among all flows. Packets **1**, **4**, **6**, and B arrive at the same time and all have greedy flow sources, i.e., all flows are heavily backlogged. By comparing the output pattern in DRR with that in WFQ shown in FIG. **4***a*, three problems may be observed. First, packets **1**, **4** and **6** were transmitted under DRR out of sequence (i.e., in comparison to WFQ such as **6**, **1** and **4**) since DRR only considers whether a packet could be sent out in a round and does not consider eligible transmission sequence for packets. Second, packets **6**, **7**, **8** and **9** are sent out in a batch under DRR, which in terms of latency and fairness is not considered good behavior in a packet switched network. Third, the transmission time of packet B (with a size slightly greater than the residual credits of the first round) is delayed under DRR until the next round, i.e., after all other flows finish their transmissions in the second round, which may be too long a delay. Under DRR, the delay increases with the frame size and a larger quantum size produces larger frame size.

FIG. **4***b* shows an embodiment consistent with the present invention using Pre-Order Deficit Round Robin and operating on the same input pattern and assumptions used in FIG. **4***a*. In this example, packets are classified into 4 classes, so that Z=4. Assuming for all flows the quantums are equal to 400 and the size of packet B is 500, then packet B cannot be sent out in the first round. However, in the next round, DeficitCounter would be equal to 300, i.e. 400+400−500. In accordance with the present invention, the packet would then be classified into the first class, i.e., 4−⌊300/(400/4)⌋ and could be sent out at the beginning of the next round. Other packets are put into the priority queuing sub-module **115** according to the same rule. Thus, as shown in FIG. **4***b*, even under the assumption that all flows are heavily backlogged and there are enough priority queues, embodiments consistent with the present invention exhibit good performance in comparison to the typical DRR performance.

Below, the performance of PDRR is analyzed in terms of delay bound and throughput fairness. Under the analysis below, PDRR is shown to have an O(1) per-packet time complexity in most case. In particular, consider a queuing system with a single server of rate C.

Definition 2: A backlogged period for flow i is any period of time during which flow i is continuously backlogged in the system. Time t_0 is the beginning of a backlogged period of flow i and t_k indicates the time that the kth round in PDRR

is completed. $W_i(\tau,t_k)$ denotes the service offered to flow i in the interval $(\tau,t_k]$ by the server and L_i is the maximum packet size of flow i.

Lemma 3: Under PDRR, if flow i is continuously backlogged in the interval $(t_0,t_k]$ then at the end of the kth round,

$$W_i(t_0,t_k) \geq k\phi_i - D_i^k, \tag{9}$$

where D_i^k is the value of the DeficitCounter$_i$ at the end of the kth round and ϕ_i is Quantum$_i$.

Lemma 4: Let t_0 be the beginning of a backlogged period of flow i in PDRR. At any time t during the backlogged period,

$$W_i(t_0,t) \geq \max\left\{0, r_i\left(t-t_0-\frac{\left(2+\frac{1}{Z}\right)F-\phi_i}{C}\right)\right\}. \tag{10}$$

where F is equal to

$$\sum_{i}^{\#flows}\phi_i,$$

Z is the number of priority queues, and r_i is the rate allocated to the flow i.

Theorem 1: The PDRR server belongs to LR with latency θ^{PDRR} less than or equal to

$$\frac{\left(2+\frac{1}{Z}\right)F-\phi_i}{C}. \tag{11}$$

According to equation (1), replacing F with $\phi_i C/r_i$ in equation (11) results in

$$\theta^{PDRR} \leq \left(2+\frac{1}{Z}\right)\frac{\phi_i}{r_i}-\frac{2\phi_i}{C}. \tag{12}$$

Equation (11) shows that PDRR improves latency, as opposed to the DRR whose latency is $(3F-2\phi_i)/C$. Furthermore, in the worst case, if the form of θ^{SCFQ} is translated to the form of θ^{PDRR}, the latency of PDRR is shown to be similar to that of SCFQ, which is $(2F-\phi_i)/C$. Equation (12) demonstrates that the latency of PDRR is inversely dependent with the allocated bandwidth, and independent of the number of active flows.

Theorem 2: The scheduling algorithm at the server is PDRR and the traffic of flow i conforms to a leaky bucket with parameters (σ_i,ρ_i), where σ_i and ρ_i denote the burstiness and average rate of the flow i, respectively. The rate allocated to the flow i is assumed to be equal to ρ_i. If Delay$_i$ is the delay of any packet of flow i, then

$$\text{Delay}_i \leq \frac{\sigma_i}{\rho_i}+\frac{\left(2+\frac{1}{Z}\right)F-\phi_i}{C}. \tag{13}$$

Theorem 3: For a PDRR scheduler,

$$\text{Fairness}^{PDRR} = \frac{\left(2+\frac{1}{Z}\right)F}{C}. \tag{14}$$

where Fairness^{PDRR} is the fairness of the server PDRR. Thus, Fairness^{PDRR} is smaller than Fairness^{DRR}, which is 3F/C.

As noted in the analysis above, for each packet, PKT_Arrival inserts the packet into its corresponding Fq and PKT_Pass takes the packet from its Fq to the Pq$_j$ where j is found in a constant number of operations. PKT_Departure repeatedly picks a packet from the $Pq_{MinHeapRoot}$ whose MinHeapRoot always

TABLE 4

THE TRAFFIC PARAMETERS AND QUANTUM SIZE OF TWO GROUPS

Group	Traffic Type	Bit Rate (Mbps)	Packet Size (byte)	Quantum Size (byte)
GA	CBR	4	50	500
GB	CBR	4	500	500

presents the smallest j among non-empty Pq$_j$'s. As the min heap operations are not invoked under the assumption that each accessed Pq is non-empty, all the complexities of the above operations are O(1). When the $Pq_{MinHeapRoot}$ is empty, a delete operation of the min heap is invoked by PKT_Departure to get the new MinHeapRoot. The reheapification loop has time complexity O(log Z), where Z is the maximum number of keys present in the "min heap", i.e., the number of non-empty Pq's at that moment. A similar situation also occurs as PKT_Pass must insert a new j into the min heap. Above, Table 4 summarizes the complexity of PKT_Arrival, PKT_Pass, PKT_Departure when a non-empty or empty priority queue is accessed.

However, embodiments consistent with the present invention allow for packets to be sent out individually and have low delay possibility by operations of the min heap. First, the min heap operation may be involved when the accessed Pq is empty. In contrast, sorted-priority algorithms, e.g., SCFQ and WFQ, use two operations, insert and delete, repeatedly when a packet is being sent out. Secondly, embodiments consistent with the present invention allow for the scalar of the min heap to be small such that the maximum number of keys is the number of Pq's instead of the number of flows. Moreover, embodiments consistent with the present invention allow concurrent insertions and deletions on the heap. According to the principles of the present invention, PKT_Departure, after getting the next smallest value j immediately via one comparison between the two leaf keys of the root, can start to send out packets in the Pq$_j$ concurrently during the period of the reheapification loop. Therefore, the time complexity of PDRR in accordance with the present invention is O(1) in most cases and O(log Z) in some special cases. Accordingly, embodiments consistent with the present invention utilize an algorithm with a lower complexity than algorithms such as SCFQ, which requires O(log N) operations where N is the number of flows.

FIGS. 5-8 show various simulation results to compare the performance of embodiments consistent with the present invention using PDRR, with DRR and SCFQ. For the

simulation, the link bandwidth, i.e., server capacity, was assumed to be 80 Mbps, shared by 20 flows. The 20 flows were divided into two groups, GA and GB. The following experiment was performed to show the problem of bursty transmission by DRR and the improvement effected by PDRR. The traffic sources were assumed to be Constant Bit Rate (CBR) with fixed packet size. Below, Table 5 indicates the traffic parameters of the two groups, GA and GB.

TABLE 5

	Operation		
Case	PKT ARRIVAL	PKT PASS	PKT DEPARTURE
Pq non-empty	O(1)	O(1)	O(1)
P empty	O(1)	O(log Z)	O(log Z)

Furthermore, the bit rates of GA and GB were set equal, the maximum packet size among both groups was 500 bytes, and the same quantum size of 500 bytes was allocated to them. The packet arrival rate of GA flows was 10 times the size of GB. In this experiment, 10 priority queues were used. The delay time of a particular GA flow under DRR, PDRR, and SCFQ, respectively were measured.

As shown in FIG. 5, packets of flow within DRR are sent out in a batch once the flow is served. In PDRR the flow is able to spend its quantum in several pieces, so that packets could be sent out uniformly. Another observation is that packets in SCFQ suffer high delay jitter, which is due to GA flows having a packet arrival rate ten times that of the GB flow. Further, for GA packets that arrive while the server is serving a large packet of GB, their virtual arrival times are equal, which causes the server not to send them out in their actual arrival sequence.

As shown in FIG. 6, the GA flow in DRR has a larger average delay than that in PDRR. The traffic source was assumed to be shaped by a leaky bucket with on/off rates are both 1000 1/μsec and a bit rate of 4 Mbps. The GA flows were assigned a larger packet size than GB flows, however all flows requested the same bandwidth. With the increase of GA's packet size to that of GB's, the curve of PDRR in FIG. 6 shows that PDRR performs well, especially with heterogeneous traffic sources.

FIG. 7 shows that as the ratio of GA's packet size to GB's packet size is increased, small packets in PDRR perform better than large packets, unlike the case of DRR. GA flows were assumed to have a higher bit rate than that of GB flows. Unlike DRR, PDRR considers the information provided in the quantum consumed by a packet and reorders the transmission sequence of packets in one round. Under DRR, a node only considers whether a packet could be sent out and ignores the transmission order of packets in one round. In a high flow environment with heterogeneous bandwidth requirements, embodiments consistent with the present invention using PDRR perform better than DRR because pre-order queuing module 108 can transmit packets more uniformly within a round. In addition, FIG. 8 shows that the average delay of GA flows in PDRR is lower than that in DRR.

FIG. 9 shows performance of an embodiment consistent with the present invention as the number of priority queues is varied for a specific traffic environment. As described above, embodiments consistent with the present invention enable a flow to use its quantum uniformly within a round, especially when its quantum is several times its maximum packet size. For example, for a flow i, pre-order queuing with (Quantum$_i$/L$_i$) priority queues can reach the above goal. Thus, for a specific traffic environment, Z priority queues are sufficient where Z is equal to the maximum value of (Quantum$_i$/L$_i$) among all flows, i.e.

$$\max_i(Quantum_i / L_i).$$

FIG. 9 shows experimental data illustrating the relationship between Z and

$$\max_i(Quantum_i / L_i).$$

All traffic sources were assumed as CBR with fixed packet size. FIG. 9 shows the average delay of the flow whose (Quantum$_i$/L$_i$) equals

$$\max_i(Quantum_i / L_i).$$

For each line, this flow receives the smallest average delay when

$$Z = \max_i(Quantum_i / L_i),$$

which confirms the advantages realized through practice of the present invention.

The present invention may also apply to flows with different allocated bandwidth but having the same distribution of packet sizes. Traffic types, other than CBR and MMPP, are also within the principles of the present invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

APPENDIX: PROOFS OF PRIMARY RESULTS

Proof of Lemma 1: This is proved by showing that for any packet P_i^m, its DeficitCounter$_i^m$ must be positive and smaller than Quantum$_i$. The value in DeficitCounter cannot be negative and could only increase by Quantum$_i$ in the UpdateOneFlow procedure. It is assumed that there is insufficient credit to send out the packet P_i^m, i.e., the DeficitCounter$_i^{m-1}$ is smaller than L_i^m, the size of this packet. After updating and sending out the packet P_i^m,

$$\text{DeficitCounter}_i^m = \text{DeficitCounter}_i^{m-1} + \text{Quantum}_i - L_i^m.$$

As DeficitCounter$_i^{m-1}$<L_i^m, DeficitCounter$_i^m$ must be smaller than Quantum$_i$. As a result, lemma 1 is proved.

Proof of lemma 2: According to equation (6), equation (5) is equivalent to

$$QA_i^m = QA_i^{m-1} - \frac{L_i^m}{\text{Quantum}_i} \quad (A.1)$$

Since $L_i^m>0$, $\text{Quantum}_i>0$ and $r_i>0$, from equation (3) and equation (A.1) for any m

$$TS_i^m > TS_i^{m-1}$$

and

$$QA_i^{m-1} > QA_i^m.$$

In the same round, for the packet P_i^m with the smallest timestamp, its m is smallest among all packets, and the QA_i^m of the packet with the smallest m. Thus, for the packet with the smallest timestamp in one round, its QA is the largest.

Proof of Lemma 3: PDRR only modifies the service sequence of the packets in a DRR round. Therefore, the packets in DRR that could be sent out in a round during a backlogged period, can still be sent out in the same PDRR round.

Proof of Lemma 4: For each time interval $(t_{k-1},t_k]$,

$$t_k - t_{k-1} \le \frac{1}{C}\left(F + \sum_{j=1}^{N} D_j^{k-1} - \sum_{j=1}^{N} D_j^k\right). \quad (A.2)$$

By summing over k−1,

$$t_{k-1} - t_0 \le (k-1)\frac{F}{C} + \frac{1}{C}\sum_{j=1}^{N} D_j^0 - \frac{1}{C}\sum_{j=1}^{N} D_j^{k-1}. \quad (A.3)$$

It is assumed there are two packets, P_i^A and P_i^B, in the Fq_i whose sizes are L_i^A and L_i^B, $(L_i^B-\phi_i)$, respectively, and only P_i^A can be sent out at the (k−1)th round. All other flows exhaust their DeficitCounter. Thus, $D_i^{K-1}=\phi_i-\Delta$ where $0<\Delta\le\phi_i$, $D_j^{k-1}=0$ for $j\neq i$, and

$$t_{k-1} - t_0 \le (k-1)\frac{F}{C} + \frac{F-\phi_i+\Delta}{C}. \quad (A.4)$$

Under this assumption, in the kth round P_i^B would be placed into the Pq_n where

$$\begin{aligned} n &= Z - \left\lfloor \frac{D_i^k}{Pq\_\phi_i} \right\rfloor \quad (A.5)\\ &= Z - \left\lfloor \frac{D_i^{k-1} + \phi_i - L_i^B}{\phi_i / Z} \right\rfloor \\ &= Z - \left\lfloor \frac{D_i^{k-1}}{\phi_i} Z \right\rfloor \\ &= Z - \left\lfloor \frac{\phi_i - \Delta}{\phi_i} Z \right\rfloor \\ &= Z - \left\lfloor Z - \frac{\Delta}{\phi_i} Z \right\rfloor \end{aligned}$$

$$= \left\lceil \frac{\Delta}{\phi_i} Z \right\rceil$$

and the maximum amount of data that could be served before P_i^B is $((n/Z)F-\phi_i)$. Thus, for any time t from the beginning of the kth round until the P_i^B is served,

$$\begin{aligned} t - t_0 &\le \frac{\frac{n}{Z}F - \phi_i}{C} + t_{k-1} - t_0 \quad (A.6)\\ &\le (k-1)\frac{F}{C} + \frac{F-\phi_i+\Delta}{C} + \frac{\frac{n}{Z}F-\phi_i}{C}, \end{aligned}$$

or equivalently,

$$k - 1 \ge \frac{(t-t_0)C}{F} + \frac{2\phi_i - \Delta}{F} - \frac{n}{Z} - 1. \quad (A.7)$$

Replacing k with k−1 in equation (9) and defining r_i, the reserved rate of flow i, as $(\phi_i C/F)$, results in

$$\begin{aligned} W_i[t_0, t_{k-1}) &\ge \phi_i\left(\frac{(t-t_0)C}{F} + \frac{2\phi_i-\Delta}{F} - \frac{n}{Z} - 1\right) - D_i^{k-1} \quad (A.8)\\ &= r_i\left(t - t_0 + \frac{2\phi_i-\Delta}{C} - \frac{n}{Z}\times\frac{F}{C} - \frac{F}{C} - \frac{D_i^{k-1}}{r_i}\right) \\ &= r_i\left(t - t_0 - \left(1 + \frac{n}{Z} + \frac{D_i^{k-1}}{\phi_i}\right)\frac{F}{C} + \frac{2\phi_i-\Delta}{C}\right) \\ &= r_i\left(t - t_0 - \frac{\left(1 + \frac{n}{Z} + \frac{D_i^{k-1}}{\phi_i}\right)F - 2\phi_i + \Delta}{C}\right) \end{aligned}$$

Replacing D_i^{k-1} with $\phi_i-\Delta$ and since

$$\frac{n}{Z} - \frac{\Delta}{\phi_i} = \left\lceil \frac{\Delta}{\phi_i} Z \right\rceil \frac{1}{Z} - \frac{\Delta}{\phi_i} \le \frac{1}{Z}$$

for any Δ_i

$$W_i[t_0, t_{k-1}) \ge r_i\left(t - t_0 - \frac{\left(2 + \frac{n}{Z} - \frac{\Delta}{\phi_i}\right)F - 2\phi_i + \Delta}{C}\right) \quad (A.9)$$

$$\ge r_i\left(t - t_0 - \frac{\left(2 + \frac{1}{Z}\right)F - \phi_i}{C}\right).$$

In the worst case, flow i was the last to be updated during the kth round and its packet L_i is inserted into the tail of Pq_n. The following two cases are considered:

Case 1: At time t before the time that flow i is served in the kth round, i.e.

$$t_{k-1} < t \le t_{k-1} + \frac{n}{Z}F - \phi_i,$$

there results

$$W_i(t_0,t) = W_i(t_0,t_{k-1}). \quad (A.10)$$

Case 2: At time t after the time that flow i starts being served in the kth round, i.e.

$$t_{k-1} + \frac{n}{Z}F - \phi_i < t \le t_k,$$

there results

$$W_i(t_0,t) = W_i(t_0,t_{k-1}) + W_i(t_{k-1},t) \ge W_i(t_0,t_{k-1}). \quad (A.11)$$

Thus, for any time t,

$$W_i(t_0,t) \ge \max\left\{0,\ r_i\left(t - t_0 - \frac{\left(2+\frac{1}{Z}\right)F - \phi_i}{C}\right)\right\}. \quad (A.12)$$

Proof of Theorem 3: The beginning of the kth round is assumed and there are two packets P_i^A and P_i^B in the Fq_i whose sizes are L_i^A and ϕ_i, respectively, and only P_i^A can be sent out at the kth round. The packet P_i^B's class is n, which implies it will enter the nth priority queue. In the (k+1) round, all packets of another flow j whose classes are larger than n, could not be sent out before the time t when P_i^B is sent out, as the server always selects packets from the noncmpty Pq_j with the smallest j. Thus, before t, for another flow j,

$$W_j(t_0,t) \le k\phi_j + \frac{n}{Z}\phi_j. \quad (A.13)$$

Also as $t_k<t<t_{k+1}$, from equation (9) for flow i,

$$W_i(t_0,t) \ge (k-1)\phi_i + \Delta. \quad (A.14)$$

From equations (A.5), (A.13) and (A.14) it is concluded that

$$\left|\frac{W_i(t_0,t)}{r_i} - \frac{W_j(t_0,t)}{r_j}\right| \le \left(k+\frac{n}{Z}\right)\frac{\phi_j}{r_j} - (k-1)\frac{\phi_i}{r_i} - \frac{\Delta}{r_i} \le \left(1+\frac{n}{Z} - \frac{\Delta}{\phi_i}\right)\frac{F}{C} \le \left(1+\frac{1}{Z}\right)\frac{F}{C}. \quad (A.15)$$

This bound applies to time intervals that began at t_0. For any arbitrary interval,

$$\left|\frac{W_i(t_1,t_2)}{r_i} - \frac{W_j(t_1,t_2)}{r_j}\right| \le \left(2+\frac{1}{Z}\right)\frac{F}{C}. \quad (A.16)$$

Thus, for any two flows i and j,

$$\text{Fairness}^{PDRR} = \frac{\left(2+\frac{1}{Z}\right)F}{C}. \quad (A.17)$$

What is claimed is:

1. A method for scheduling a packet, comprising the steps of:
   - receiving a packet having a size;
   - identifying a flow for said packet by at least a flow identifier;
   - classifying said packet based on said identified flow;
   - buffering said packet in one of a plurality of queues, arranged in a priority order, based on said classification of said packet and a priority of said packet assigned based on said priority order;
   - allocating a predetermined amount of bandwidth to said identified flow;
   - determining an accumulated bandwidth based on said predetermined amount of bandwidth; and
   - processing said packet in the one of the plurality of queues based on said accumulated bandwidth and said size of said packet.

2. The method of claim 1, wherein identifying said flow for said packet comprises identifying a source address of said packet.

3. The method of claim 1, wherein identifying said flow for said packet comprises identifying a destination address of said packet.

4. The method of claim 1, wherein classifying said packet comprises:
   - calculating said size of said packet; and
   - calculating said accumulated bandwidth assigned to said flow based upon said size of said packet.

5. The method of claim 4, wherein calculating said accumulated bandwidth is based upon predetermined amount of bandwidth assigned to said flow and said size of said packet.

6. The method of claim 1, wherein buffering said packet in one of said plurality of queues comprises:
   - arranging said plurality of queues in a priority order;
   - assigning a priority to said packet based on said priority order; and
   - buffering said packet in one of said queues based on said assigned priority.

7. The method of claim 6, wherein assigning a priority to said packet based on said priority order comprises;
   - determining said size of said packet; and
   - calculating a transmission delay based on said size of said packet and said priority order.

8. The method according to claim 1, further including:
   - calculating a residual bandwidth after said processing;
   - allocating a second predetermined amount of bandwidth to said identified flow; and
   - recalculating said accumulated bandwidth based on said residual bandwidth and said second predetermined amount of bandwidth.

9. The method according to claim 8, wherein:
   - said residual bandwidth is determined as a difference between said accumulated bandwidth and a total amount of data of packets processed; and
   - said accumulated bandwidth is recalculated as a summation of said residual bandwidth and snid second predetermined amount of bandwidth.

10. A system for scheduling a packet, comprising:
an input to receive a plurality of packets;
an arrival module to identify a flow for each of said plurality of packets by at least a flow identifier;
a classifier to assign each of said plurality of packets to one of a plurality of queues, arranged in a priority order, based on said identified flow;
a server for allocating a predetermined amount of bandwidth to said identified flow, determining an accumulated bandwidth based on said predetermined amount of bandwidth, and selecting one of said plurality of queues based on said priority order; and
an output for outputting a packet from said selected queue based on said accumulated bandwidth of said identified flows, a priority of said packet assigned based on said priority order, and said size of said packet.

11. The system of claim 10, further comprising:
a memory to store a service list of flows identified for each of said plurality of packets.

12. An apparatus for scheduling a packet, comprising:
means for receiving a packet having a size;
means for identifying a flow for said packet by at least a flow identifier;
means for classifying said packet based on said identified flow;
means for buffering said packet in one of a plurality of queues, arranged in a priority order, based on said classification of said packet and a priority of said packet assigned based on said priority order;
means for allocating a predetermined amount of bandwidth to said identified flow;
means for determining an accumulated bandwidth based on said predetermined amount of bandwidth; and
means for processing said packet in the one of the plurality of queues based on the accumulated bandwidth and said size of said packet.

13. A program on a computer readable medium for configuring a processor to execute a method for scheduling a packet, said method comprising the steps of:
receiving a packet having a size;
identifying a flow for said packet by at least a flow identifier;
classifying said packet based on said identified flow;
buffering said packet in one of a plurality of queues, arranged in a priority order, based on said classification of said packet and a priority of said packet assigned based on said priority order;
allocating a predetermined amount of bandwidth to said identified flow;
determining an accumulated bandwidth based on said predetermined amount of bandwidth; and
processing said packet in the one of the plurality of queues based on said accumulated bandwidth and said size of said packet.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,236,491 B2
APPLICATION NO. : 09/955296
DATED : June 26, 2007
INVENTOR(S) : Shih-Chiang Tsao et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 19, line 15, "flows", should read --flow--.

Signed and Sealed this

Twenty-first Day of August, 2007

JON W. DUDAS
Director of the United States Patent and Trademark Office

UNITED STATES DISTRICT COURT
CENTRAL DISTRICT OF CALIFORNIA

NOTICE OF ASSIGNMENT TO UNITED STATES MAGISTRATE JUDGE FOR DISCOVERY

This case has been assigned to District Judge Josephine Tucker and the assigned discovery Magistrate Judge is Arthur Nakazato.

The case number on all documents filed with the Court should read as follows:

SACV11- 1378 JST (ANx)

Pursuant to General Order 05-07 of the United States District Court for the Central District of California, the Magistrate Judge has been designated to hear discovery related motions.

All discovery related motions should be noticed on the calendar of the Magistrate Judge

==========

NOTICE TO COUNSEL

A copy of this notice must be served with the summons and complaint on all defendants (if a removal action is filed, a copy of this notice must be served on all plaintiffs).

Subsequent documents must be filed at the following location:

[ ] **Western Division**
312 N. Spring St., Rm. G-8
Los Angeles, CA 90012

[X] **Southern Division**
411 West Fourth St., Rm. 1-053
Santa Ana, CA 92701-4516

[ ] **Eastern Division**
3470 Twelfth St., Rm. 134
Riverside, CA 92501

Failure to file at the proper location will result in your documents being returned to you.

COPY

Mark A. Flagel
Ryan E. Hatch
Jessica C. Kronstadt
LATHAM & WATKINS LLP
355 South Grand Avenue, Los Angeles, California 90071-1560
Tel: (213) 485-1234

Dean G. Dinduras
LATHAM & WATKINS LLP
650 Town Center Drive - 20th Floor, Costa Mesa, CA 92626-1925
Tel: (714) 540-1235

UNITED STATES DISTRICT COURT
CENTRAL DISTRICT OF CALIFORNIA

A10 NETWORKS, INC., a Delaware Corporation,

PLAINTIFF(S)

v.

BROCADE COMMUNICATIONS SYSTEMS, INC., a Delaware Corporation, and F5 NETWORKS, INC., a Washington Corporation,

DEFENDANT(S).

CASE NUMBER

SACV11-01378 JST (ANx)

SUMMONS

TO: DEFENDANT(S): _____

A lawsuit has been filed against you.

Within 21 days after service of this summons on you (not counting the day you received it), you must serve on the plaintiff an answer to the attached ☑ complaint ☐ _____ amended complaint ☐ counterclaim ☐ cross-claim or a motion under Rule 12 of the Federal Rules of Civil Procedure. The answer or motion must be served on the plaintiff's attorney, Mark Flagel, whose address is LATHAM & WATKINS LLP, 355 South Grand Avenue, Los Angeles, California 90071. If you fail to do so, judgment by default will be entered against you for the relief demanded in the complaint. You also must file your answer or motion with the court.

Clerk, U.S. District Court

Dated: SEP - 9 2011

By: SUSANA P. BUSTAMANTE
Deputy Clerk

SEAL

(Seal of the Court)

[Use 60 days if the defendant is the United States or a United States agency, or is an officer or employee of the United States. Allowed 60 days by Rule 12(a)(3)].

UNITED STATES DISTRICT COURT, CENTRAL DISTRICT OF CALIFORNIA
CIVIL COVER SHEET

I (a) PLAINTIFFS (Check box if you are representing yourself ☐)
A10 Networks, Inc.,
a Delaware Corporation

DEFENDANTS
Brocade Communications Systems, Inc., a Delaware Corporation, and
F5 Networks, Inc., a Washington Corporation

(b) Attorneys (Firm Name, Address and Telephone Number. If you are representing yourself, provide same.)
Mark A. Flagel, Ryan E. Hatch, Jessica C. Kronstadt
Latham & Watkins LLP, 355 South Grand Avenue, Los Angeles, CA 90071, Tel. 213.485.1234
Dean G. Dunlavey
Latham & Watkins LLP, 650 Town Center Drive - 20th Floor, Costa Mesa, CA 92626, Tel. 714.540.1235

Attorneys (If Known)

II. BASIS OF JURISDICTION (Place an X in one box only.)

☐ 1 U.S. Government Plaintiff ☑ 3 Federal Question (U.S. Government Not a Party)

☐ 2 U.S. Government Defendant ☐ 4 Diversity (Indicate Citizenship of Parties in Item III)

III. CITIZENSHIP OF PRINCIPAL PARTIES - For Diversity Cases Only
(Place an X in one box for plaintiff and one for defendant.)

	PTF	DEF		PTF	DEF
Citizen of This State	☐ 1	☐ 1	Incorporated or Principal Place of Business in this State	☐ 4	☐ 4
Citizen of Another State	☐ 2	☐ 2	Incorporated and Principal Place of Business in Another State	☐ 5	☐ 5
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6

IV. ORIGIN (Place an X in one box only.)

☑ 1 Original Proceeding ☐ 2 Removed from State Court ☐ 3 Remanded from Appellate Court ☐ 4 Reinstated or Reopened ☐ 5 Transferred from another district (specify): ☐ 6 Multi-District Litigation ☐ 7 Appeal to District Judge from Magistrate Judge

V. REQUESTED IN COMPLAINT **JURY DEMAND:** ☑ Yes ☐ No (Check 'Yes' only if demanded in complaint.)

CLASS ACTION under F.R.C.P. 23: ☐ Yes ☑ No ☐ **MONEY DEMANDED IN COMPLAINT: $**

VI. CAUSE OF ACTION (Cite the U.S. Civil Statute under which you are filing and write a brief statement of cause. Do not cite jurisdictional statutes unless diversity.)
35 U.S.C. § 100 et seq.

VII. NATURE OF SUIT (Place an X in one box only.)

OTHER STATUTES
☐ 400 State Reapportionment
☐ 410 Antitrust
☐ 430 Banks and Banking
☐ 450 Commerce/ICC Rates/etc.
☐ 460 Deportation
☐ 470 Racketeer Influenced and Corrupt Organizations
☐ 480 Consumer Credit
☐ 490 Cable/Sat TV
☐ 810 Selective Service
☐ 850 Securities/Commodities/ Exchange
☐ 875 Customer Challenge 12 USC 3410
☐ 890 Other Statutory Actions
☐ 891 Agricultural Act
☐ 892 Economic Stabilization Act
☐ 893 Environmental Matters
☐ 894 Energy Allocation Act
☐ 895 Freedom of Info. Act
☐ 900 Appeal of Fee Determination Under Equal Access to Justice
☐ 950 Constitutionality of State Statutes

CONTRACT
☐ 110 Insurance
☐ 120 Marine
☐ 130 Miller Act
☐ 140 Negotiable Instrument
☐ 150 Recovery of Overpayment & Enforcement of Judgment
☐ 151 Medicare Act
☐ 152 Recovery of Defaulted Student Loan (Excl. Veterans)
☐ 153 Recovery of Overpayment of Veteran's Benefits
☐ 160 Stockholders' Suits
☐ 190 Other Contract
☐ 195 Contract Product Liability
☐ 196 Franchise

REAL PROPERTY
☐ 210 Land Condemnation
☐ 220 Foreclosure
☐ 230 Rent Lease & Ejectment
☐ 240 Torts to Land
☐ 245 Tort Product Liability
☐ 290 All Other Real Property

TORTS
PERSONAL INJURY
☐ 310 Airplane
☐ 315 Airplane Product Liability
☐ 320 Assault, Libel & Slander
☐ 330 Fed. Employers' Liability
☐ 340 Marine
☐ 345 Marine Product Liability
☐ 350 Motor Vehicle
☐ 355 Motor Vehicle Product Liability
☐ 360 Other Personal Injury
☐ 362 Personal Injury-Med Malpractice
☐ 365 Personal Injury-Product Liability
☐ 368 Asbestos Personal Injury Product Liability

IMMIGRATION
☐ 462 Naturalization Application
☐ 463 Habeas Corpus-Alien Detainee
☐ 465 Other Immigration Actions

TORTS
PERSONAL PROPERTY
☐ 370 Other Fraud
☐ 371 Truth in Lending
☐ 380 Other Personal Property Damage
☐ 385 Property Damage Product Liability

BANKRUPTCY
☐ 422 Appeal 28 USC 158
☐ 423 Withdrawal 28 USC 157

CIVIL RIGHTS
☐ 441 Voting
☐ 442 Employment
☐ 443 Housing/Accommodations
☐ 444 Welfare
☐ 445 American with Disabilities - Employment
☐ 446 American with Disabilities - Other
☐ 440 Other Civil Rights

PRISONER PETITIONS
☐ 510 Motions to Vacate Sentence Habeas Corpus
☐ 530 General
☐ 535 Death Penalty
☐ 540 Mandamus/ Other
☐ 550 Civil Rights
☐ 555 Prison Condition

FORFEITURE / PENALTY
☐ 610 Agriculture
☐ 620 Other Food & Drug
☐ 625 Drug Related Seizure of Property 21 USC 881
☐ 630 Liquor Laws
☐ 640 R.R. & Truck
☐ 650 Airline Regs
☐ 660 Occupational Safety /Health
☐ 690 Other

LABOR
☐ 710 Fair Labor Standards Act
☐ 720 Labor/Mgmt. Relations
☐ 730 Labor/Mgmt. Reporting & Disclosure Act
☐ 740 Railway Labor Act
☐ 790 Other Labor Litigation
☐ 791 Empl. Ret. Inc. Security Act

PROPERTY RIGHTS
☐ 820 Copyrights
☑ 830 Patent
☐ 840 Trademark

SOCIAL SECURITY
☐ 861 HIA (1395ff)
☐ 862 Black Lung (923)
☐ 863 DIWC/DIWW (405(g))
☐ 864 SSID Title XVI
☐ 865 RSI (405(g))

FEDERAL TAX SUITS
☐ 870 Taxes (U.S. Plaintiff or Defendant)
☐ 871 IRS-Third Party 26 USC 7609

FOR OFFICE USE ONLY: Case Number: SACV11-01378

AFTER COMPLETING THE FRONT SIDE OF FORM CV-71, COMPLETE THE INFORMATION REQUESTED BELOW.

UNITED STATES DISTRICT COURT, CENTRAL DISTRICT OF CALIFORNIA
CIVIL COVER SHEET

VIII(a). IDENTICAL CASES: Has this action been previously filed in this court and dismissed, remanded or closed? ☑ No ☐ Yes
If yes, list case number(s):

VIII(b). RELATED CASES: Have any cases been previously filed in this court that are related to the present case? ☑ No ☐ Yes
If yes, list case number(s)

Civil cases are deemed related if a previously filed case and the present case:
(Check all boxes that apply) ☐ A. Arise from the same or closely related transactions, happenings, or events; or
☐ B. Call for determination of the same or substantially related or similar questions of law and fact; or
☐ C. For other reasons would entail substantial duplication of labor if heard by different judges; or
☐ D. Involve the same patent, trademark or copyright, and one of the factors identified above in a, b or c also is present.

IX. VENUE: (When completing the following information, use an additional sheet if necessary.)

(a) List the County in this District; California County outside of this District; State if other than California; or Foreign Country, in which **EACH** named plaintiff resides.
☐ Check here if the government, its agencies or employees is a named plaintiff. If this box is checked, go to item (b).

County in this District:*	California County outside of this District; State, if other than California; or Foreign Country
Orange County	Santa Clara County

(b) List the County in this District; California County outside of this District; State if other than California; or Foreign Country, in which **EACH** named defendant resides.
☐ Check here if the government, its agencies or employees is a named defendant. If this box is checked, go to item (c).

County in this District:*	California County outside of this District; State, if other than California; or Foreign Country
Orange County (Brocade Communications Systems, Inc.; F5 Networks, Inc.)	Santa Clara County (Brocade Communications Systems, Inc.; F5 Networks, Inc.)

(c) List the County in this District; California County outside of this District; State if other than California; or Foreign Country, in which **EACH** claim arose.
Note: In land condemnation cases, use the location of the tract of land involved.

County in this District:*	California County outside of this District; State, if other than California; or Foreign Country
Orange County Los Angeles County	Santa Clara County

*** Los Angeles, Orange, San Bernardino, Riverside, Ventura, Santa Barbara, or San Luis Obispo Counties**
Note: In land condemnation cases, use the location of the tract of land involved

X. SIGNATURE OF ATTORNEY (OR PRO PER): _____ Date September 9, 2011

Notice to Counsel/Parties: The CV-71 (JS-44) Civil Cover Sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law. This form, approved by the Judicial Conference of the United States in September 1974, is required pursuant to Local Rule 3-1 is not filed but is used by the Clerk of the Court for the purpose of statistics, venue and initiating the civil docket sheet. (For more detailed instructions, see separate instructions sheet.)

Key to Statistical codes relating to Social Security Cases:

Nature of Suit Code	Abbreviation	Substantive Statement of Cause of Action
861	HIA	All claims for health insurance benefits (Medicare) under Title 18, Part A, of the Social Security Act, as amended. Also, include claims by hospitals, skilled nursing facilities, etc., for certification as providers of services under the program. (42 U.S.C. 1935FF(b))
862	BL	All claims for "Black Lung" benefits under Title 4, Part B, of the Federal Coal Mine Health and Safety Act of 1969. (30 U.S.C. 923)
863	DIWC	All claims filed by insured workers for disability insurance benefits under Title 2 of the Social Security Act, as amended; plus all claims filed for child's insurance benefits based on disability. (42 U.S.C. 405(g))
863	DIWW	All claims filed for widows or widowers insurance benefits based on disability under Title 2 of the Social Security Act, as amended. (42 U.S.C. 405(g))
864	SSID	All claims for supplemental security income payments based upon disability filed under Title 16 of the Social Security Act, as amended.
865	RSI	All claims for retirement (old age) and survivors benefits under Title 2 of the Social Security Act, as amended. (42 U.S.C. (g))